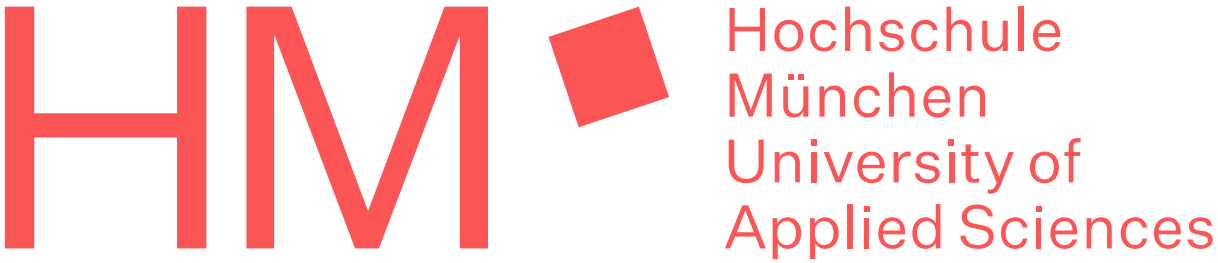


Bachelorarbeit

Detektion von Einzelbäumen in Luftbildern mit Hilfe eines neuronalen Netzes unter Berücksichtigung von Abweichungen in der Radiometrie der Bilddaten

angefertigt von

Marina Trinz

Fakultät für Geoinformation
Studiengang Geoinformatik und Navigation

Sommersemester 2022

vorgelegt bei

Prof. Dr.-Ing. Andreas Schmitt

München, den 05.05.2022

Kurzfassung

Kenntnisse über die Anzahl der Bäume in Wäldern und Städten sind wichtig, sowohl aus wirtschaftlicher als auch aus ökologischer Sicht. Durch ein einheitliches und ständig aktualisiertes Baumkataster kann die Überwachung des Vegetationszustandes zugänglicher gestaltet werden. Mit Methoden des maschinellen Lernens können kosten- und zeitaufwändige traditionelle Vorgänge für die Kartierung einzelner Bäume optimiert und automatisiert werden. Die Aufgabe der Detektion einzelner Bäume in hochaufgelösten Fernerkundungsdaten wird in diesem Projekt mit den Methoden aus dem Computer Vision Bereich angegangen.
Die Objektdetektion war immer eine der wichtigsten Aufgaben von Computer Vision und lässt sich heutzutage sehr gut mit Hilfe von neuronalen Netzen lösen. Die für das Projekt Einzelbaumdetektion durchgeführten Experimente zeigen, dass auch im Bereich der Fernerkundung mit sogenannten Deep-Learning-basierten Objekterkennungsmethoden gute Ergebnisse erzielt werden können. Das neuronale Netz YOLOv3 hat bei der Einzelbaumdetektion in diesem Projekt eine mittlere durchschnittliche Präzision von 0,869 auf dem Testdatensatz erreicht.
Um die Robustheit von diesem Ergebnis zu untersuchen, wird dessen Übertragbarkeit in Bezug auf Abweichungen in der Radiometrie der Luftbilder ausgewertet. Solchen Abweichungen liegen Unterschiede bei den Erfassungsmethoden, bei den atmosphärischen Bedingungen und bei den Beleuchtungsstärken zugrunde. Als Referenz bei diesen Experimenten wird die K-Neares-Neighbors-Klassifikation der untersuchten Gebiete verwendet. So wird bewertet, wie sich das Detektionsergebnis in Bezug auf mögliche Abweichungen in der Radiometrie der Luftbilder verändert.
Außerdem wird in diesem Projekt eine Methode für die Anpassung der radiometrischen Abweichungen mittels unüberwachter Domänenanpassung mit dem Netz CycleGAN erprobt. Die durchgeführten Experimente haben gezeigt, dass durch diese Anpassung eine Verbesserung des Detektionsergebnisses erreicht werden kann. Bei den erfolgreich angepassten Untersuchungsgebieten konnte sowohl eine Vergrößerung des Anteils der detektierten Vegetation an der Gesamtvegetation als auch eine Minimierung der Fehldetektionen beobachtet werden.

Das allgemeine Ziel dieses Projektes ist, zu bewerten, inwieweit sich die Deep-Learning-basierte Objekterkennungsmethode für die Erstellung eines Baumkatasters eignet und welche Aspekte dabei berücksichtigt werden müssen.

Abstract

Knowledge about the number of trees in forests and cities is important, both from an economic and ecological point of view. Monitoring the state of vegetation can be made more accessible through a uniform and constantly updated tree register. With machine learning methods, costly and time-consuming traditional processes for mapping single trees can be optimized and automated. The task of detecting single trees in very high resolution remote sensing data is tackled in this project using computer vision technique for object detection.
Object detection has always been one of the most important tasks in computer vision and can now be solved very well with the help of neural networks. The experiments carried out for the single tree detection project show that good results can also be achieved in the field of remote sensing with so-called deep learning-based object recognition methods. The neural network YOLOv3 achieved a mean average precision of 0.869 on the test dataset for single tree detection in this project.
In order to examine the robustness of this result, its transferability in relation to deviations in the radiometry of the aerial photographs is evaluated. Underlying such discrepancies are differences in detection methods, atmospheric conditions, and illuminance levels. The K-Neares-Neighbors classification of the surveyed areas is used as a reference in these experiments. In this way, it is evaluated how the detection result changes in relation to possible deviations in the radiometry of the aerial photos.
In addition, a method for the adjustment of the radiometric deviations by means of unsupervised domain adaptation with the network CycleGAN is tested in this project. The experiments carried out have shown that this adaptation can improve the detection result. In the successfully adapted surveyed areas, both an increase in the proportion of detected vegetation to the total vegetation and the minimization of false detections could be observed.

The goal of this project is to evaluate to what extent the deep learning based object detection method is suitable for the creation of a tree register and which aspects have to be considered.

Danksagung

Ich möchte mich zunächst bei allen bedanken, die mich während meiner Bachelorarbeit unterstützten.

Dazu gehören mein Betreuer Herr Prof. Dr. Andreas Schmitt von der Hochschule München und meine Betreuerinnen Isabell Riesinger und Bettina Henjes vom Landesamt für Digitalisierung, Breitband und Vermessung.

Für die Organisation dieses außergewöhnlichen Forschungsprojekts und für die Bereitstellung der Fernerkundungsdaten, welche für die Berechnungen in diesem Projekt benutzt wurden, bedanke ich mich hiermit beim gesamten Referat 85 des Landesamtes für Digitalisierung, Breitband und Vermessung.

Ich möchte mich an dieser Stelle auch bei meinem Mann Sebastian Mades bedanken, der mich während meines Studiums begleitete und mich bedingungslos unterstützte.

Vielen Dank.

Erklärung

gemäß § 16 Abs. 10 APO in Zusammenhang mit § 35 Abs. 7 RaPO

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Der Freistaat Bayern besitzt laut der Bundeswaldinventur aus dem Jahr 2012 rund 2,6 Millionen Hektar Wald (LWF, 2012). Durch den Klimawandel wird eine erhöhte Kontrolle der Vegetation sowohl in den Wäldern als auch in den Städten erforderlich, denn nicht alle Bäume können warmes und trockenes Wetter vertragen.
Für die Überwachung des Vegetationszustandes und der Artenvielfalt der Bäume wird ein komplettes, einheitliches und ständig aktualisiertes Baumkataster gebraucht. Diese Informationsquelle kann sowohl für eine ökologische Stadtplanung als auch für die Verbesserung der Waldnutzung aus wirtschaftlicher Sicht genutzt werden. Zu diesem Zweck müssen kostspielige und zeitaufwändige Felduntersuchungen zur Bewertung der Vegetationsressourcen durch neue Methoden der Datengewinnung und Datenauswertung ersetzt werden.
Fernerkundungsdaten und Methoden des maschinellen Lernens werden schon seit Jahren erfolgreich für die Beobachtung und Kartierung der Vegetation eingesetzt (Petersen u. a., 2008).
In einem Konferenzbeitrag des Deutschen Fernerkundungsdatenzentrums am DLR wurden Klassifikationsergebnisse des Random Forest Verfahrens (Breiman, 2001) für die Kartierung der Grünflächen in deutschen Städten präsentiert. Für die Klassifikation wurden Sentinel-2 Aufnahmen (ESA, 2022) verwendet. Es wurden die Klassen Laubbäume, Nadelbäume, Sträucher und Grasflächen ermittelt. Dabei wurde eine Gesamtgenauigkeit von 92,3% erreicht (Tenikl u. a., 2019).
Das Random Forest Verfahren wurde für die Ermittlung des Alters der Sekundärwälder im brasilianischen Amazonasgebiet aus Daten von Landsat 5 TM und ALOS-PALSAR (JAXA, 2022) eingesetzt (Carreiras u. a., 2017).
Die Kombination aus K-Nearest-Neighbors (Diawara, 2019) und einem geführten Filter (He u. a., 2013) ermöglichte eine spektral-räumliche Klassifikation eines Kiefernwaldes im Nordwesten von Indiana, USA (Guo u. a., 2018). Als Datengrundlage dienten multispektrale Aufnahmen des AVIRIS-Sensors (NASA, 2022).
Eine Studie der Georg-August-Universität Göttingen hat gezeigt, dass K-Nearest-Neighbors für das Abschätzen der Biomasse einzelner Bäume verwendet werden kann (Fehrmann und Kleinn, 2007).
Obwohl man bei der Klassifikation der Vegetationsfläche große Erfolge erzielt hat, bleibt die Aufgabe der Einzelbaumdetektion ein aktives Forschungsthema. Besonders in Bereichen mit einer hohen Baumdichte, wie es in Parks und Wäldern der Fall ist, ist die Einzelbaumdetektion eine Herausforderung. Das hängt damit zusammen, dass die Grenzen eines Baums schwer zu definieren sind.
Dank der Entwicklung zahlreicher Objekterkennungsmethoden mit Hilfe von neuronalen Netzen wird der Einsatz vieler derartiger Methoden auch in der Fernerkundung möglich.
Für die Überwachung gesetzlich geschützter Baumarten in Brasilien wurden Ergebnisse der neuronalen Netze Faster R-CNN (Ren u. a., 2015), YOLOv3 (Redmon und Farhadi, 2018) und RetinaNet (Lin u. a., 2017b) verglichen. Mit nur drei Spektralkanälen (Rot, Grün und Blau) haben alle drei Netze eine mittlere durchschnittliche Präzision von über 80% bei der Detektion erreicht (Santos u. a., 2019).
Das neuronale Netz RetinaNet hat eine mittlere durchschnittliche Präzision von 86,1% bei der

Detektion von Phoenix-Palmen gezeigt und bietet damit eine Methode zur Kartierung dieser Bäume, welche einen geringeren Arbeitsaufwand im Vergleich zu einer Messkampagne vor Ort hat (Culman u. a., 2020).
In einer Pilotstudie der Technischen Universität Dresden wurde ein neuronales Netz zur Detektion von Stadtgrün in hochaufgelösten Luftbildern mit einer Bodenpixelgröße von 20 cm eingesetzt. Die Detektion wurde mit fünf Baumarten durchgeführt, wobei jede Baumart eine eigene Klasse repräsentiert. Außer der Spektralinformation aus dem Grün-Kanal wurde der normalisierte Vegetationsindex (NDVI) und das normalisierte digitale Oberflächenmodell (nDOM) verwendet (Haas, 2019).

Das in dieser Arbeit beschriebene Projekt der Einzelbaumdetektion am Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) wird durch die oben genannte Pilotstudie der TU Dresden inspiriert. Die Fernerkundungsdaten Digitale True Orthophotos, Digitale Geländemodelle und Digitale Oberflächenmodelle, welche vom LDBV produziert werden, ermöglichen die Verwendung ähnlicher Methoden.
In diesem Projekt werden Experimente mit dem neuronalen Netz YOLOv3 und dem neuronalen Netz RetinaNet durchgeführt. Die Ergebnisse, die mit RetinaNet erzielt werden, dienen als Vergleich für die Ergebnisse, die mit YOLOv3 erreicht werden. Das Ziel dieser Experimente ist, zu überprüfen, wie gut sich diese neuronalen Netze für die Einzelbaumdetektion sowohl in Wäldern als auch in Städten eignen.
Außer der Anwendung von Validierungs- und Testdatensätzen wird in diesem Projekt eine weitere Bewertungsmethode der Detektionsergebnisse vorgeschlagen. Dazu werden die Detektionsergebnisse mit den Ergebnissen der flächendeckenden K-Nearest-Neighbors-Klassifikation der Untersuchungsgebiete verglichen. Diese Methode bietet die Möglichkeit, unabhängig von einem Testdatensatz das Ergebnis der Detektionsmodelle zu bewerten. Dies ist wichtig, da die Datensätze für das Trainieren und das Testen der Netze für die Objektdetektion schwer zu erstellen sind. Aus diesem Grund enthalten sie oft nicht genug unterschiedliche Objektbeispiele für die Erstellung eines Objektdetektionsmodells. Infolgedessen kann es dazu kommen, dass dieses Objektdetektionsmodell nicht alle Objekte in einem Bild detektiert. Durch den Vergleich mit einer flächendeckenden Klassifikation kann der Anteil der detektierten Vegetation an der Gesamtvegetation bestimmt werden.
Zusätzlich wird überprüft, wie robust die vortrainierten Detektionsmodelle gegen Abweichungen in der Radiometrie der Bilddaten sind. Solche Abweichungen können z.B. auf Grund von Unterschieden bei den Erfassungsmethoden, bei den atmosphärischen Bedingungen und bei den Beleuchtungsstärken auftreten. Bei der Erfassung der Vegetation spielt außerdem der Aufnahmezeitraum eine große Rolle, da der Vegetationszustand nicht über das ganze Jahr hinweg konstant bleibt.
Für die Erhöhung der Robustheit der Detektionsmodelle wird in diesem Projekt die unüberwachte Domänenanpassung in Betracht gezogen. Das Ziel der Domänenanpassung ist die Anpassung einer Verteilung an eine weitere Verteilung. Man spricht dabei von den Quell- und Zieldomänen. Diese Methode gewinnt an Bedeutung im Fernerkundungsbereich, da sie keinen zusätzlichen Aufwand bei der Datenaufbereitung erfordert.
Huang u. a. (2019) haben das Problem von unterschiedlichen Aufnahmesensoren und die damit verbundenen radiometrischen Abweichungen mit der unüberwachten Domänenanpassung angegangen.
Benjdira u. a. (2019) haben mittels unüberwachter Domänenanpassung die Spektralkanäle der Luftbilder Nahinfrarot, Rot und Grün in die Spektralkanäle Rot, Grün und Blau umgewandelt.

Damit wurde die Segmentierungsgenauigkeit des auf der Quelldomäne vortrainierten Modells angewendet auf die Zieldomäne von 14% auf 61% erhöht.
In diesem Projekt wird untersucht, wie gut sich das neuronale Netz CycleGAN (Zhu u. a., 2017) für die unüberwachte Domänenanpassung im Fernerkundungsbereich eignet.

2 Theoretische Grundlagen

In diesem Kapitel werden die allgemeinen theoretischen Grundlagen erläutert, die für das Verstehen des Aufbaus und der Auswertung der einzelnen Experimente in diesem Projekt nötig sind.

2.1 Künstliche neuronale Netze

Künstliche neuronale Netze sind komplexe Algorithmen, die zum Bereich des überwachten maschinellen Lernens gehören. Maschinelles Lernen ist ein Sammelbegriff für verschiedene statistische Methoden, welche in Form von zusammengesetzten Algorithmen für das Bearbeiten von großen Datensätzen und die anschließende Informationsextraktion eingesetzt werden (Gareth u. a., 2013). Ein wesentlicher Aspekt, der diese Algorithmen auszeichnet, ist, dass die oben genannten Aufgaben von den Algorithmen selbständig abgearbeitet werden. Somit kann die Analyse von großen und vielschichtigen Datenmengen effizient gestaltet werden.
Das allgemeine Ziel von überwachtem Lernen ist, eine Funktion f zu finden, die den Zusammenhang zwischen der Menge der Datenpunkte $\mathcal{X}$ und der dazugehörigen Menge der Zielwerte $\mathcal{Y}$ eines Datensatzes möglichst gut repräsentiert:

$$f : \mathcal{X} \to \mathcal{Y} \tag{2.1}$$

Die Güte der erstellten Funktion wird mit Hilfe eines unabhängigen Testdatensatzes verifiziert. Dabei wird die Funktion auf die Menge der Datenpunkte $\mathcal{X}_{test}$ angewendet und das Ergebnis mit den Zielwerten $\mathcal{Y}_{test}$ verglichen. Dadurch kann überprüft werden, ob sich die Funktion nur für die Beschreibung eines Datensatzes eignet oder ob sie auch auf ähnliche Analysenaufgaben übertragbar ist.

2.1.1 Grundlagen

Im Folgenden werden vereinfacht der Aufbau und die Funktionsweise eines neuronalen Netzes erklärt.

Aufbau eines neuronalen Netzes

Der kleinste Baustein jedes neuronalen Netzes ist ein Neuron. Ein Neuron ist eine Funktion, welche die Entscheidungsgrenze bei einer Klassifikation repräsentiert. Die Entscheidungsgrenze wird durch die Gewichte $\mathbf{w}$ und den Bias-Parameter w_0 definiert. Die Dimension des Gewichtsvektors $\mathbf{w}$ ist durch die Anzahl der Merkmale der Datenpunkte bestimmt. Bei einem mehrdimensionalen Merkmalsraum $\mathbb{R}^n$ stellt sich die Entscheidungsgrenze als eine komplexe Hyperebene dar.

Die Parameter $\mathbf{w}$ und w_0 der Funktion, die die Entscheidungsgrenze beschreibt, werden anhand der einzelnen Merkmale $\mathbf{x}$ der Datenpunkte $\mathcal{X}$ iterativ angepasst. Mit dieser Anpassung wird angestrebt, dass die Funktionsergebnisse $f(\mathbf{x})$ mit möglichst vielen Zielwerten $\mathcal{Y}$ übereinstimmen. Durch die Anwendung einer Aktivierungsfunktion α auf Zwischenwerte z innerhalb der Funktion f hat das Funktionsergebnis $f(\mathbf{x})$ den gleichen Wertebereich wie die Zielwerte $\mathcal{Y}$. Das Funktionsergebnis wird auch als Vorhersage des erstellten Modells bezeichnet und ist für einen Datenpunkt wie folgt definiert:

$$f(\mathbf{x}) = \alpha(\mathrm{z}), \tag{2.2}$$

wobei

$$\mathrm{z} = \sum_{i=1}^{n} \mathrm{w}_i \cdot \mathrm{x}_i + \mathrm{w}_0, \text{ mit } \mathbf{w} \in \mathbb{R}^n \text{ und } \mathbf{x} \in \mathbb{R}^n \tag{2.3}$$

Diese komplexen Zusammenhänge sind in der Abbildung 2.1 schematisch dargestellt.

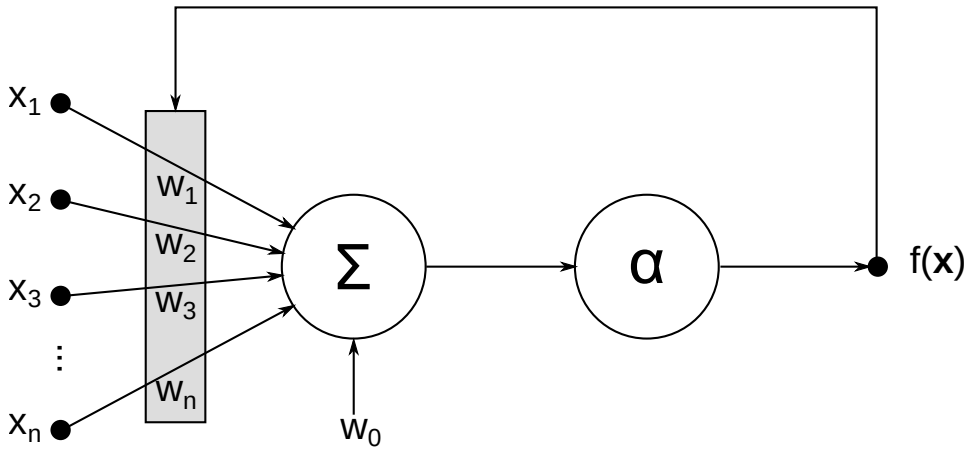


Abbildung 2.1: Eine schematische Darstellung eines Neurons. Die direkte Verbindung zwischen dem Funktionsergebnis $f(\mathbf{x})$ und dem Gewichtsvektor $\mathbf{w}$ bedeutet eine Aktualisierung der Gewichte in Abhängigkeit von den Ausgabewerten.

Ein einzelnes Neuron hat den Nachteil, dass es nur für die Abbildung von linearen Zusammenhängen geeignet ist. Jedoch können mit einer Gruppe von zusammenhängenden Neuronen auch nicht lineare Zusammenhänge beschrieben werden.
Eine Gruppe aus verbundenen Neuronen wird als neuronales Netz bezeichnet. In einem neuronalen Netz sind die Neuronen in Schichten aufgeteilt. Dabei unterscheidet man zwischen der Eingabeschicht, der Ausgabeschicht und den versteckten Schichten. Die Anzahl der versteckten Schichten in einem Netz ist beliebig. Dabei ist jedes Neuron der nächsten Schicht mit der kompletten Ausgabe der vorherigen Schicht verbunden. Deswegen werden solche Schichten auch vollständig verbundene Schichten (Fully Connected Layer) genannt. Ein Beispiel eines neuronalen Netzes mit vollständig verbundenen Schichten ist in der Abbildung 2.2 dargestellt.

Die nicht linearen Zusammenhänge werden mit Hilfe bestimmter Aktivierungsfunktionen α modelliert. In den neuronalen Netzen, die in diesem Projekt verwendet werden, wurden die Aktivierungsfunktionen ReLU, Leaky ReLU und Sigmoid eingesetzt.
ReLU steht für Rectified Linear Unit. Bei negativen Eingabewerten gibt diese Aktivierungsfunktion den Wert 0 zurück. Für positive Eingabewerte entspricht die Aktivierungsfunktion der Identitätsfunktion (vgl. 2.4). Der Ausgabewert 0 bedeutet, dass keine Information aus dem Datenpunkt extrahiert werden konnte und das Neuron nicht aktiviert wurde. Bei allen anderen Ausgabewerten hingegen kann eine Vorhersage getroffen werden.

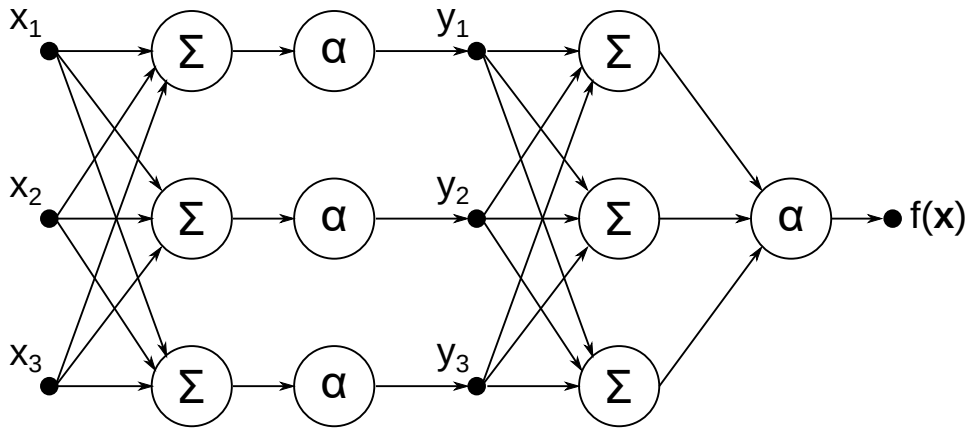


Abbildung 2.2: Ein beispielhafter Aufbau eines neuronalen Netzes mit jeweils einer Eingabeschicht, einer Ausgabeschicht und einer versteckten vollständig verbundenen Schicht. In der Ausgabeschicht wird die gesamte Information aggregiert und durch eine Aktivierungsfunktion für die Vorhersage eines Zielwertes verwendet.

Um fehlende Information zu vermeiden, wurde Leaky ReLU als weitere Variante der ReLU-Funktion vorgeschlagen. Bei negativen Eingabewerten wird statt 0 ein kleiner negativer Wert zurückgegeben (Foster, 2019). Dieser Wert wird aus dem Produkt des Eingabewertes und einem Fallkoeffizient a berechnet (vgl. 2.5). In der Bibliothek PyTorch ist der Fallkoeffizient standardmäßig auf 0,2 eingestellt (LeakyReLU, 2022).

$$\alpha_{ReLU}(\mathrm{z}) = max(0,\ \mathrm{z}) \tag{2.4}$$

$$\alpha_{LeakyReLU}(\mathrm{z}) = max(a \cdot \mathrm{z},\ \mathrm{z}) \tag{2.5}$$

Die oben beschriebenen Aktivierungsfunktionen werden üblicherweise in den versteckten Schichten der Netze für die Verknüpfung zwischen den Schichten verwendet.
Im Vergleich zu den ReLU-Aktivierungsfunktionen, welche die Klassenzugehörigkeit ausgeben, gibt die Aktivierungsfunktion Sigmoid (vgl. 2.6) die Wahrscheinlichkeit einer Klassenzugehörigkeit aus. Diese Eigenschaft ist wichtig, wenn die Robustheit des erstellten Modells bewertet werden

muss. Deswegen wird diese Aktivierungsfunktion meistens in der letzten Schicht der Netze für die Berechnung der Vorhersage eingesetzt.

$$\alpha_{Sigmoid}(\mathrm{z}) = \frac{1}{1 + e^{-\mathrm{z}}} \tag{2.6}$$

Lernalgorithmus

Das Lernen oder das Trainieren eines neuronalen Netzes bedeutet eine iterative Anpassung der Gewichte $\mathbf{w}$ mit dem Aktualisierungsparameter $\Delta\mathbf{w}$, so dass mit der Entscheidungsgrenze die Zielwerte, auch Klassen genannt, eines Datensatzes möglichst gut beschrieben werden können. Dafür wird eine Fehlerfunktion E definiert, mit der die Abweichung zwischen dem Zielwert $\mathbf{y}$ und der Vorhersage $\hat{\mathbf{y}}$ gemessen wird. Im Laufe des Lernens muss die Fehlerfunktion minimiert werden. Dieses Verfahren wird Gradientenabstiegsverfahren genannt, da das Minimum durch die partielle Ableitung der Fehlerfunktion $\nabla E(\mathbf{w})$ nach den Gewichten angenähert wird.
Die Berechnung des Aktualisierungsparameters $\Delta\mathbf{w}$ wird in der Regel nur mit einem Teil des kompletten Datensatzes (Batch) durchgeführt. Der Grund dafür ist die Rechenzeit und die damit verbundenen Rechenkosten, die durch die Nutzung eines Teildatensatzes verringert werden. Der Nachteil dieser Methode ist, dass die zufällig ausgewählten Teildatensätze die Bewertung des Endergebnisses und die anschließende Anpassung der Gewichte zusätzlich verrauschen können.
Ein wichtiger Parameter des Gradientenabstiegsverfahrens ist die Lernrate η. Die Lernrate gibt an, wie stark die Gewichte innerhalb einer Lerniteration t verändert werden. Die Beziehung zwischen der Lernrate und der Anpassung der Gewichte wird wie folgt beschrieben:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta\Delta\mathbf{w}_t, \tag{2.7}$$

mit

$$\Delta\mathbf{w}_t = \nabla E(\mathbf{w}_t) \tag{2.8}$$

Bei der Wahl der Lernrate ist zu beachten, dass sowohl zu kleine als auch zu große Werte das Konvergieren des Lernens verhindern können. An den Stellen, an denen die Fehlerfunktion ein flaches Plateau hat, wird eine kleine Lernrate bedeuten, dass viele Lerniterationen gebraucht werden, um das Plateau zu verlassen. Wiederum besteht bei einer großen Lernrate die Gefahr, dass es zu Oszillationen kommt und das Minimum nicht erreicht wird. Die Lösung besteht darin, eine adaptive Lernrate zu verwenden, die im Laufe des Lernens aktualisiert wird.
Für die Optimierung des Gradientenabstiegsverfahren in Bezug auf die adaptive Lernrate wurden verschiedene Techniken entwickelt. Besonders verbreitete Optimierungstechniken sind das Stochastic Gradient Descent (SGD) mit dem Momentum-Hyperparameter und das Adaptive Moment Estimation (Adam).
Mit der SGD-Optimierung wird ein gewichteter Mittelwert der Gradienten aus früheren Lerniterationen berechnet, mit dem die Entwicklung der Lernrate gesteuert wird. Die Adam-Optimierung bestimmt die optimale Lernrate durch das Verhältnis der exponentiellen Glättung der Gradienten

zur Wurzel aus der exponentiellen Glättung der Gradientenquadraten.
Das Ziel der beiden Techniken ist, dadurch die Entwicklung der Gradienten zu beobachten, um z.B. in der Nähe eines Funktionsminimums die Lernrate zu reduzieren und damit das Konvergieren zu beschleunigen (Ketkar und Moolayil, 2021).
Die Anzahl der Iterationen, die für das Konvergieren des Lernens und für die Erstellung eines Modells benötigt werden, kann nicht im Voraus bestimmt werden. Aus diesem Grund wird vor dem Lernen ein Abbruchkriterium festgelegt. Dadurch wird definiert, wie viele Iterationen die Anpassung der Gewichte dauern soll. Als Abbruchkriterium kann entweder eine bestimmte Anzahl von Lerniterationen oder eine Anzahl an Iterationen ohne eine erhebliche Verbesserung der Fehlerfunktion ausgewählt werden. Die zweite Methode verhindert unnötige Lerniterationen und ist kosteneffizienter.
Wie die Gewichte in der ersten Lerniteration initialisiert sind, hat eine große Auswirkung auf deren weitere Anpassung während des Lernens. Die Gewichtsinitialisierung bleibt ein Forschungsthema und es gibt nur wenige Erkenntnisse darüber, wie sich die Initialisierung auf die Dauer und das Konvergieren des Lernens auswirkt.
Glorot und Bengio (2010) zeigen, dass die verwendete Aktivierungsfunktion und die Anzahl der Gewichte bei der Initialisierung der Gewichte berücksichtigt werden müssen.
Inzwischen gibt es eine allgemeine Regel, die sich etabliert hat. Laut dieser Regel ist es wichtig, die Gewichte mit zufälligen Zahlen zu initialisieren. Die Verwendung von gleichen Gewichten zusammen mit gleichen Aktivierungsfunktionen am Anfang des Lernens führt dazu, dass gleiche Gradienten und gleiche Anpassungen berechnet werden. Das führt zur sogenannten Symmetrie der Gewichte. Dies wiederum kann das Konvergieren des Lernens verhindern.
Die restlichen Netzparameter werden im weiteren Verlauf durch das Bearbeiten der Eingangsdaten berechnet. Diese Berechnung beginnt an der Eingabeschicht. Im Gegensatz dazu wird die Anpassung der Gewichte an der Ausgabeschicht begonnen. Dabei wird die Ableitung der Fehlerfunktion rekursiv durch alle Netzschichten nach der Kettenregel berechnet. Dieser Vorgang wird als Rückwärtspropagierung bezeichnet.
Auf diesem Prinzip basiert auch das Lernverfahren von Convolutional Neural Networks, welche im nächste Unterkapitel beschrieben werden.

2.1.2 Convolutional Neural Network (CNN)

CNNs sind tiefe neuronale Netze, da sie aus mehreren versteckten Schichten bestehen. Diese Netze werden schon seit langem erfolgreich für die Bildklassifikation in der Computer Vision eingesetzt. Die Idee von solchen Netzen wurde in (LeCun u. a., 1989) vorgestellt.
Im Unterschied zu anderen Klassifikationsmethoden, welche die Bildklassifikation nur pixelweise durchführen können, sind CNNs in der Lage die räumlichen Objektmerkmale aus den Bildern zu extrahieren. Die Bilder werden schließlich anhand von diesen Objektmerkmalen klassifiziert. Für die Extraktion der Merkmale verfügen die CNNs über besondere versteckte Faltungsschichten (Convolutional Layer), die durch vollständig verbundene Schichten für die Klassifikationsaufgabe gestützt werden.
In den Faltungsschichten werden per Faltungsoperation die Eingabedaten mit den Merkmalsfiltern gefaltet (Goodfellow u. a., 2016). Im Gegensatz zu vollständig verbundenen Schichten ist ein einzelnes Neuron in einer Faltungsschicht nur mit einem Teil der Eingabe verbunden. Die Größe dieses Teilbereichs der Eingabedaten ist für jedes Neuron durch die Größe des Merkmalsfilters bestimmt.
Ein Merkmalsfilter ist ein Operator, mit dem ein bestimmtes Muster in den Eingabedaten

gefunden werden kann. In den tieferen Schichten des Netzes wird zuerst nach einfachen Mustern wie z.B. Kanten gesucht. Eine Faltung s von den Eingabedaten x mit einem Merkmalsfilter k im Abschnitt t kann wie folgt beschrieben werden:

$$s(t) = (x * k)(t) \tag{2.9}$$

In Abbildung 2.3 ist ein Beispiel für die Berechnung der Faltung innerhalb einer Faltungsschicht zu sehen. Dabei werden auf die Eingabe mit der Größe $I \times I \times 1$ drei unterschiedliche Merkmalsfilter angewendet. Bei einer Eingabe mit der Größe $I \times I \times n$ würde jeder Merkmalsfilter auf jeden der n Kanäle angewendet.

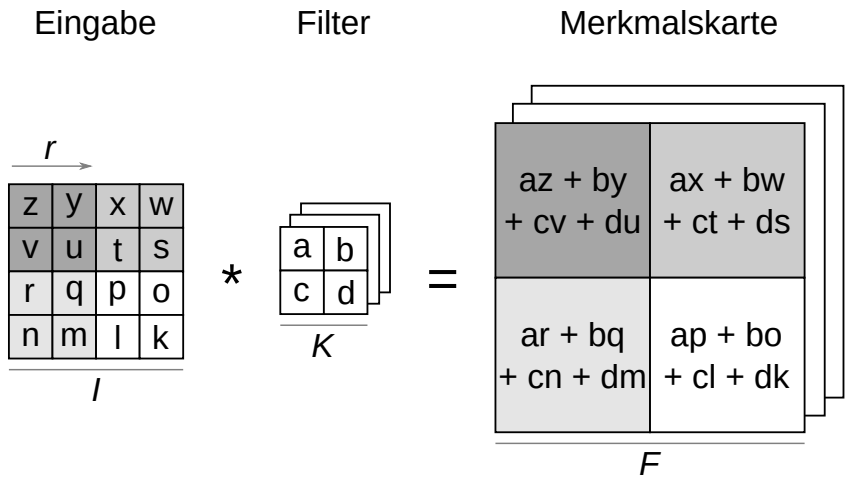


Abbildung 2.3: Berechnung der Faltung der Eingabe mit der Größe $I \times I$ mit drei Merkmalsfiltern mit der Größe $K \times K$, wo $I = 4$ und $K = 2$. Der Filterabstand r ist auf 2 gesetzt.

Alle Merkmalsfilter, die für die Berechnung der Faltung verwendet werden, werden von dem Netz während des Lernens entwickelt. Die Verwendung von vordefinierten Merkmalsfiltern wird nicht mehr praktiziert. Jeder Filter wird für die Extraktion eines bestimmten Merkmals eingesetzt.
Das Ergebnis der Faltung ist eine Merkmalskarte, die auch als Feature Map bezeichnet wird. Die Auflösung der Merkmalskarte F hängt von der Auflösung der Eingabedaten I, der Filtergröße K und dem Abstand r, in dem der Filter angewendet wird, wie folgt ab:

$$F = \frac{I - K}{r} + 1 \tag{2.10}$$

Für das obere Bildbeispiel entspricht die Größe der Merkmalskarte $2 \times 2 \times 3$. Somit ist sie viel kleiner als die Größe der Eingabe. Die Größe der dritten Dimension ist von der Anzahl der angewendeten Filter abhängig.
Dieses Beispiel zeigt, wie mit der Faltung eine Reduktion der Datenauflösung erreicht werden kann. Die Datenauflösung hat eine direkte Auswirkung auf die Anzahl der Netzparameter in jeder Schicht. Durch die Anpassung der Filtergröße und des Filterabstands lässt sich die Parameteranzahl eines Netzes optimieren.

Falls die Größe der Eingabe nicht reduziert werden soll, wird die Eingabe vor der Faltung künstlich vergrößert. Bei der Vergrößerung wird die Eingabe durch Nullwerte ergänzt.
Eine zusätzliche Reduktion der Datenauflösung kann auch mittels Pooling-Schichten durchgeführt werden. In diesen Schichten werden gleich große rechteckige nicht überlappende Abschnitte einer Merkmalskarte zu einzelnen Werten zusammengefasst (Weidman, 2019). Die Zusammenfassung erfolgt durch die Abbildung eines Abschnittes entweder auf den Maximalwert oder auf den Durchschnittswert dieses Abschnittes (vgl. Abbildung 2.4).

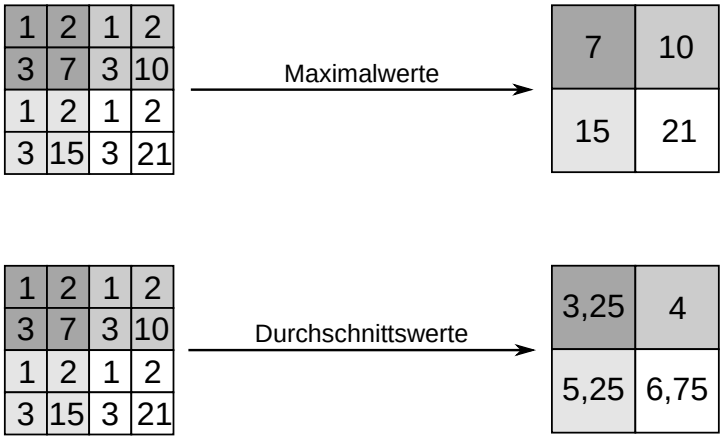


Abbildung 2.4: Ein Beispiel der möglichen Größenreduktionen einer Merkmalskarte in der Pooling-Schicht.

Nach der Merkmalsextraktion über mehrere Faltungsschichten wird die aggregierte Information in Form eines Vektors für die anschliessende Klassifikation in der vollständig verbundenen Schicht eingesetzt.
Das erste CNN namens LeNet-5, welches in (LeCun u. a., 1998) präsentiert wurde, basiert auf den oben beschriebenen Verfahren. Es wurde entwickelt, um die Zahlen zwischen 0 und 9 in Graustufenbildern mit einer Auflösung von 32×32 zu erkennen. Das Netz besteht insgesamt aus 7 Schichten (vgl. Abbildung 2.5). In der Ausgabeschicht wird die Wahrscheinlichkeit über die Klassenzugehörigkeit mit einer radialen Basisfunktion berechnet.

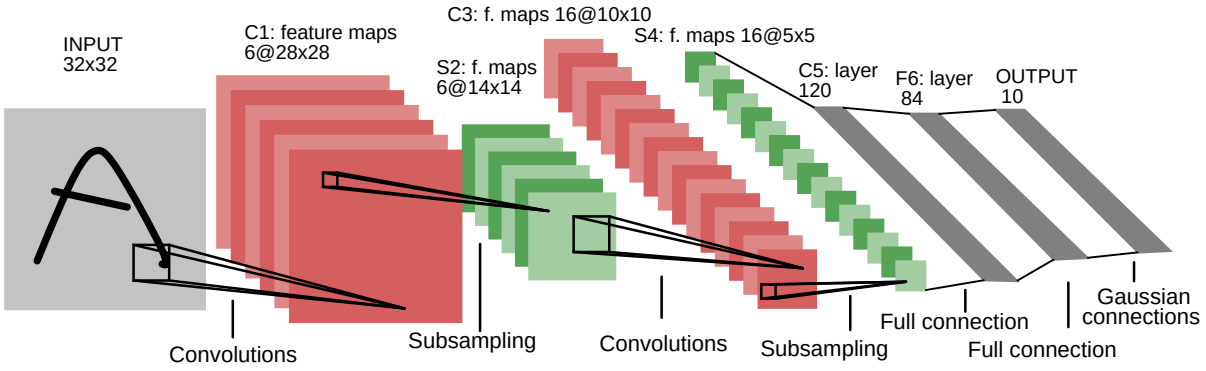


Abbildung 2.5: LeNet-5 - das erste CNN. Auf jede Faltungsschicht (rosa) mit jeweils 6 und 16 Filtern folgt eine Pooling-Schicht (grün) (Quelle: LeCun u. a. 1998, modifiziert).

Die Rechendauer und das Endergebnis eines CNN können positiv durch folgende Optimierungs-

methoden beeinflusst werden:

Vortrainierte Gewichte Für die Gewichtsinitialisierung eines CNN können auch die Gewichte verwendet werden, die im gleichen Netz für ein ähnliches Optimierungsproblem angepasst wurden. Dies ist möglich, da einige der Filter die allgemeinen Merkmale von Objekten erkennen. Die Filter, die in den ersten Faltungsschichten vortrainiert werden, eignen sich z.B. für die Extraktion einfacher Merkmale wie Kanten und Ecken.
Durch die Verwendung von vortrainierten Gewichten kann die Rechenzeit minimiert werden. Für die Erstellung der vortrainierten Gewichte kann man entweder einen eigenen Datensatz oder die öffentlich zugänglichen Datensätze verwenden. Die größten Datensätze im Bereich Bildklassifikation sind ImageNet (Russakovsky u. a., 2015) und CIFAR (Krizhevsky u. a., 2009). Für die CNNs, welche für die Objektdetektion verwendet werden, wird oft der Datensatz COCO (Lin u. a., 2014) eingesetzt. Alternativ gibt es die Möglichkeit eine Datei mit vortrainierten Gewichten herunterzuladen, welche mit den oben genannten Datensätzen erstellt wurden.

Datenerweiterung Datenerweiterung oder Data Augmentation ist eine Technik, die für die Vergrößerung der Varianz der Eingabedaten verwendet wird. Dabei werden einzelne Bilder auf unterschiedliche Weise manipuliert. Die am häufigsten verwendeten Manipulationstechniken sind die Rotation, die Skalierung und die Kontrastveränderung. Durch die Vergrößerung der Varianz der Eingabedaten wird versucht, das erstellte Modell zu generalisieren, um damit die Robustheit gegenüber Abweichungen zwischen den Trainings- und Testdatensätzen zu erzielen.

2.1.3 Generative Adversarial Networks (GAN)

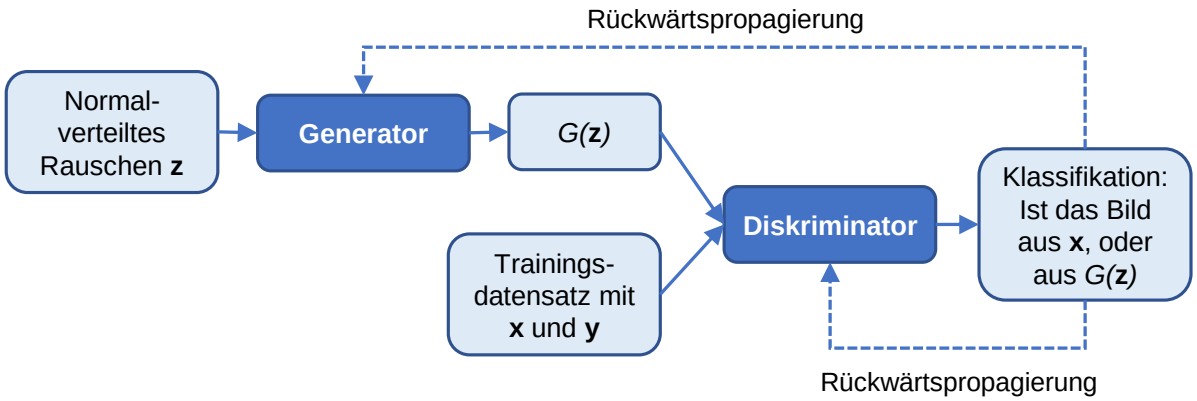


Abbildung 2.6: Die Funktionsweise eines GAN.

GANs sind neuronale Netze, die für die Generierung neuer Daten mit einer bestimmten Verteilung aus normalverteiltem Rauschen verwendet werden. Solche Netze bestehen aus zwei gegeneinander aufgestellten Netzkomponenten, einem Diskriminator und einem Generator (Goodfellow u. a., 2014). Der Generator definiert die Funktion G, welche ein normalverteiltes Rauschen $\mathbf{z}$ aus der

Menge $\mathcal{Z}$ auf die Datenpunkte $\mathbf{x}$ aus der Menge der Datenpunkte $\mathcal{X}$ abbildet:

$$G : \mathcal{Z} \to \mathcal{X} \tag{2.11}$$

Dabei hat das Rauschen $\mathbf{z}$ die Verteilung $p_{\mathbf{z}}(\mathbf{z})$ und die Datenpunkte $\mathbf{x}$ die Verteilung $p_{data}(\mathbf{x})$. Der Diskriminator ist ein Klassifikator, welcher durch die Funktion D definiert ist. Die Ausgabe des Diskriminator ist ein Wahrscheinlichkeitswert, der angibt, ob ein Beispiel aus $\mathbf{x}$ oder $G(\mathbf{z})$ stammt (vgl. Abbildung 2.6).
Das Lernziel des Diskriminators D besteht darin, seine Klassifikationsgenauigkeit zu maximieren. Wiederum ist das Lernziel des Generators, die Klassifikationsgenauigkeit von D durch die Anpassung von $G(\mathbf{z})$ auf $\mathbf{x}$ zu minimieren. Der allgemeine Lernalgorithmus eines GAN wird mit folgender Wertfunktion $V(D, G)$, bei der $\mathbb{E}$ die Fehlerfunktion ist, beschrieben:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[logD(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[log(1 - D(G(\mathbf{z})))] \tag{2.12}$$

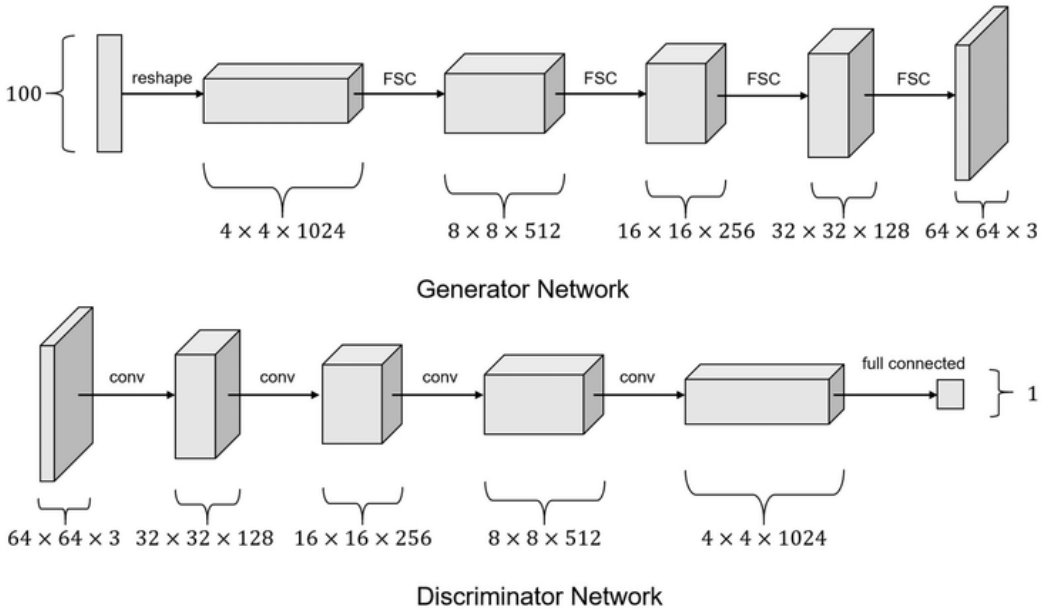


Abbildung 2.7: Die Architektur eines GAN für die Generierung neue Bilddaten am Beispiel des Netzes DCGAN (Quelle: Zhang u. a. 2020).

Bei GANs, die für die Bearbeitung von Bilddaten eingesetzt werden, wird sowohl der Generator als auch der Diskriminator durch ein CNN repräsentiert. Ein Beispiel für ein solches Netz ist das Deep Convolutional Generative Adversarial Network (DCGAN). Der Diskriminator in diesem Netz ist ein herkömmliches CNN, welches anhand eines Trainingsdatensatzes ein Klassifikationsmodell erstellt. Der Generator bekommt als Eingabe nur ein normalverteiltes Rauschen $\mathbf{z}$. Auf dieses Rauschen werden in umgekehrter Reihenfolge die gleichen Merkmalsfilter wie im Diskriminator angewendet (Radford u. a., 2015). Die Ausgabe des Generators ist ein Bild, das die gleiche Auflösung hat wie die Eingabe des Diskriminators (vgl. Abbildung 2.7). Das

Lernziel des DCGAN besteht darin, die Merkmalsfilter des Generators so anzupassen, dass die generierten Ausgabebilder $G(\mathbf{z})$ vom Diskriminator als reale Bilder klassifiziert werden könnten. Allgemein betrachtet muss dafür die Distanz zwischen der Verteilung der Eingabedaten des Diskriminators $p_{data}(\mathbf{x})$ und der Verteilung der Ausgabedaten des Generators $p_g(G(\mathbf{z}))$ minimiert werden:

$$\min_G \text{dist}(p_{data}(\mathbf{x}), p_g(G(\mathbf{z}))) \tag{2.13}$$

Eine wichtige Erweiterung der GANs bildet das Conditional GAN (cGAN). In einem cGAN wird der Prozess der Generierung neuer Daten durch eine zusätzliche Information $\mathbf{y}$ gesteuert (Mirza und Osindero, 2014). Diese Information kann z.B. die Klassenzugehörigkeit der Datenpunkte oder Datenpunkte mit einer anderen Verteilung sein. Sowohl der Generator als auch der Diskriminator haben während des Lernens Zugang zu dieser Information. Somit bildet die Generator-Funktion G nicht nur ein normalverteiltes Rauschen $\mathbf{z}$, sondern auch die zusätzliche Information $\mathbf{y}$ der Menge $\mathcal{Y}$ auf die Datenpunkte $\mathbf{x}$ ab. Der Lernalgorithmus eines cGAN wird wie folgt definiert:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[logD(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[log(1 - D(G(\mathbf{z}|\mathbf{y})))] \tag{2.14}$$

Ein möglicher Anwendungsfall der cGANs ist eine Übersetzung zwischen Bildern (vgl. Abbildung 2.8). Diese Möglichkeit wurde z.B. im neuronalen Netz pix2pix implementiert (Isola u. a., 2017).

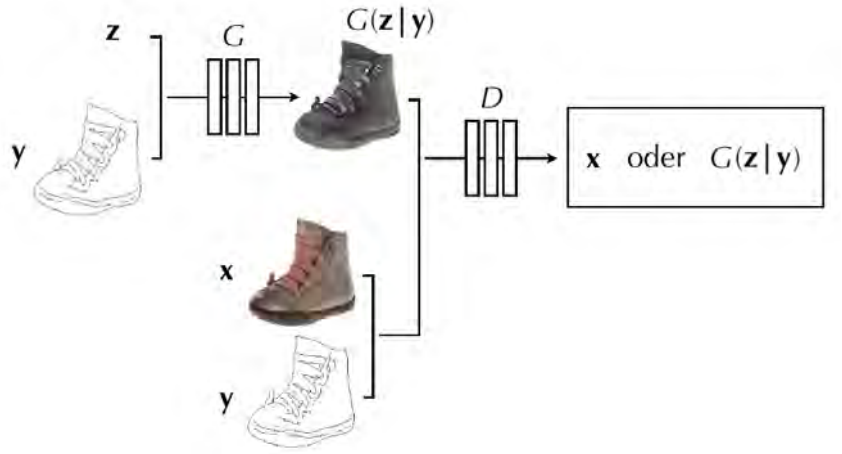


Abbildung 2.8: Die Funktionsweise des neuronalen Netzes pix2pix. Das Netz wird auf den Bildpaaren mit unterschiedlichen Verteilungen (Quell- und Zieldomäne) trainiert. Der Generator lernt Bilder aus der Quelldomäne zu generieren (Quelle: Isola u. a. 2017, modifiziert).

Allgemein ist das Ergebnis des Trainings eines GAN schwer zu bewerten. Da das Netz aus zwei gegeneinander aufgestellten Komponenten besteht, würde das Konvergieren einen Aufbau des Nash-Gleichgewichts zwischen diesen Komponenten bedeuten (Salimans u. a., 2016). Diese Aufgabe kann nicht durch das Minimieren der Fehlerfunktion mittels Gradientenabstiegsverfahren

erfüllt werden. Daher ist die Beobachtung der Entwicklung der Fehlerwerte während des Lernens nicht aussagekräftig.
Die verlässlichste und immer noch am weitesten verbreitete Methode, das Ergebnis eines GAN qualitativ zu bewerten, ist die visuelle Betrachtung der Ergebnisse durch einen Mensch.
Für die quantitative Auswertung der Ergebnisse haben sich einige Metriken wie die Fréchet Inception Distance (FID) etabliert. Für die Berechnung wird ein vortrainiertes CNN verwendet. Dieses Netz extrahiert die Merkmale der realen und generierten Bilder. Die FID berechnet sich aus den Verteilungen der extrahierten Merkmale (Heusel u. a., 2017).

2.2 Objektdetektion

Bei der Klassifikation besteht die Aufgabe darin, für ein Bild die Klassenzugehörigkeit zu bestimmen. Im Unterschied dazu wird bei der Objektdetektion jedes Objekt, das einer vordefinierten Klasse angehört, in einem Bild sowohl klassifiziert, als auch seine Position und Größe im Bild ermittelt.
Für eine richtige Klassifikation durch herkömmliche CNNs darf jedes Bild nur Objekte einer Klasse enthalten. Bei der Objektdetektion fällt diese Voraussetzung weg. Die Klassifikation erfolgt nicht mehr bildweise, sondern bereichsweise. Damit darf jedes Bild mehrere Objekte unterschiedlicher Klassen enthalten. Dadurch kann auch die Frage beantwortet werden, wieviele Objekte sich in einem Bild befinden.
Die Positionen der Objekte im Bild werden meistens in Form von Objektrahmen (Bounding Box) dargestellt. Für die Objektdetektion sind CNNs allein nicht ausreichend, sie benötigen eine Erweiterung, die den Suchprozess eines Objektes im Bild steuert. Somit wird ein Objektdetektor aus einem CNN für die Merkmalsextraktion und Erweiterungen für die Objektsuche zusammengesetzt. Ein CNN in einem Detektionsalgorithmus wird auch als Gerüst (Backbone) des Detektors bezeichnet.

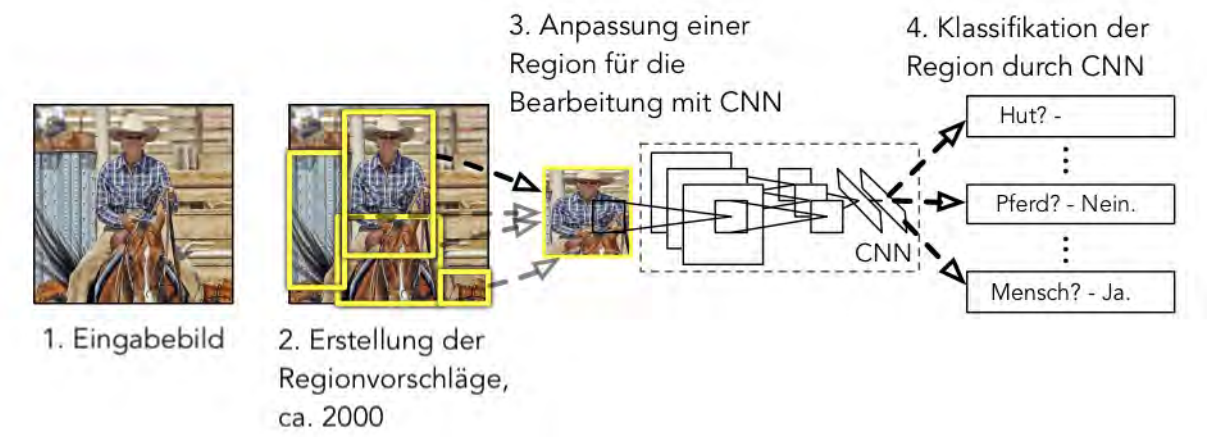


Abbildung 2.9: Die Funktionsweise von R-CNN. Ein Eingabebild wird in ca. 2000 Regionen aufgeteilt. Jede dieser Regionen wird durch ein CNN einzeln klassifiziert (Quelle: Girshick u. a. 2014, modifiziert).

Es gibt zwei Arten von neuronalen Netzen, die für Detektionsaufgaben eingesetzt werden. Sie unterscheiden sich durch den Aufbau des Suchprozesses. Sie lassen sich in zweistufige (Two-Stage

Detector) und einstufige (Single Stage Detector) Detektoren unterteilen (Jiang u. a., 2019).
Bei einem zweistufigen Detektor wird die Detektion in zwei Schritte unterteilt. Im ersten Schritt werden die Objekte in einem Bild lokalisiert, indem das Bild in Objektsegmente unterteilt wird. Die Klassifikation der Objektsegmente wird im zweiten Schritt durchgeführt (vgl. Abbildung 2.9). Dabei werden die Segmente entweder als Objekt oder als Hintergrund klassifiziert.
Obwohl mit zweistufigen Detektoren ein gutes Detektionsergebnis erreicht werden kann, erfordert die Klassifikation jedes Bildsegmentes viel Rechenleistung und ist somit relativ langsam (Redmon u. a., 2016). Ein klassisches Beispiel für einen zweistufigen Detektor ist das R-CNN (Region Based Convolutional Neural Network (Girshick u. a., 2014)) und seine Nachfolger Fast R-CNN und Faster R-CNN.
Ein einstufiger Detektor führt die Lokalisierung und die Klassifikation der Objekte in einem Schritt aus. Damit wird eine effiziente Bearbeitung der Information durch das Netz erreicht.
Zu den einstufigen Detektoren zählen die neuronalen Netze RetinaNet (Lin u. a., 2017b), Single-Shot-Detector (Liu u. a., 2016) und YOLO (Redmon u. a., 2016).

2.3 Domänenanpassung

Besonders im Fernerkundungsbereich wird das Problem unterschiedlicher Domänen des Trainings- und des Testdatensatzes zu einem Hindernis bei der automatischen Auswertung der Daten. Diese Unterschiede treten wegen Abweichungen bei den Erfassungsmethoden, bei den atmosphärischen Bedingungen und bei den Beleuchtungsstärken auf. Somit wird es erschwert, ein Modell, das mit Daten einer Domäne erstellt wurde, auf Daten einer anderen Domäne anzuwenden.

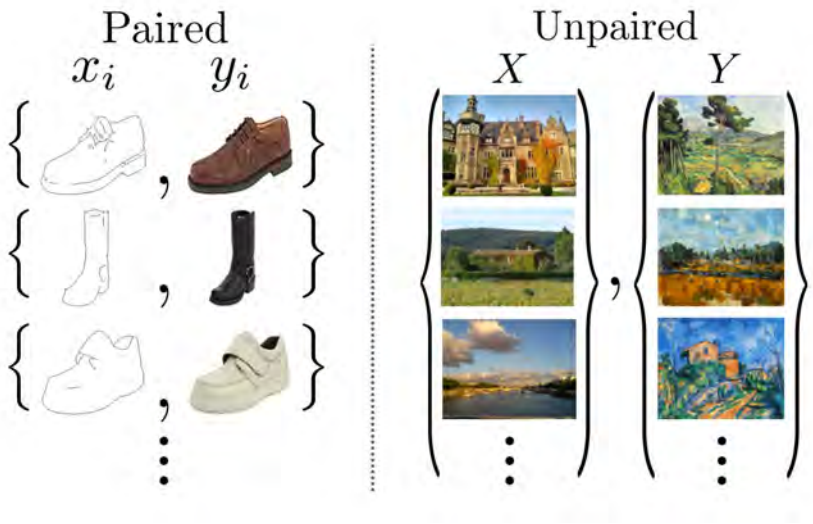


Abbildung 2.10: Der Datensatz für die gepaarte Bildübersetzung (links) mit einer eindeutigen Zuordnung zwischen Datenpunkten aus $\mathbf{x}$ und $\mathbf{y}$ und der Datensatz für die ungepaarte Bildübersetzung (rechts) ohne Verbindungen zwischen den Datenpunkten beider Domänen (Quelle: Zhu u. a. 2017).

Eine mögliche Lösung dieses Problems ist die Domänenanpassung (Domain Adaptation). Dabei wird die Verteilung der Zieldomäne der Testdaten auf die Verteilung der Quelldomäne der

Trainingsdaten angepasst.
Die Domänenanpassung kann mittels Übersetzung zwischen den Bildern realisiert werden. Ein Beispiel für eine solche Übersetzung wurde anhand des Netzes pix2pix im Abschnitt 2.1.3 erläutert. Für das Trainieren dieses Netzes wird ein Datensatz gebraucht, welcher Bildpaare der Ziel- und der Quelldomäne enthält. Das Erstellen solcher Datensätze ist sehr zeit- und kostenaufwendig. Die Alternative dazu ist die ungepaarte Bildübersetzung. Die Trainingsdatensätze für die ungepaarte Bildübersetzung enthalten wie bei der gepaarten Bildübersetzung Bilder der Ziel- und Quelldomäne, allerdings sind diese Bilder nicht mehr zu einem Paar gekoppelt (vgl. Abbildung 2.10). Daher wird die ungepaarte Bildübersetzung auch als unüberwachte Methode bezeichnet (Zhu u. a., 2017).
Eine wichtige Annahme bei der ungepaarten Bildübersetzung ist eine gewisse Ähnlichkeit zwischen der Verteilung der Zieldomäne und der Verteilung der Quelldomäne. Wie im Beispiel in Abbildung 2.10 dargestellt, handelt es sich sowohl in der Zieldomäne als auch in der Quelldomäne um Landschaftsbilder. Diese liegen in einem Fall in Form von Fotos und im anderen Fall in Form von Gemälden vor.

2.4 Validierungsmetriken

In diesem Abschnitt werden die Metriken beschrieben, die der Bewertung der Klassifikationsergebnisse dienen. Für die Erklärung der Metriken müssen die Begriffe *true positiv* (TP), *true negativ* (TN), *false positiv* (FP) und *false negativ* (FN) eingeführt werden. Die Begriffe *true positiv* und *true negativ* beschreiben den richtig klassifizierten Anteil eines Datensatzes, wohingegen die Begriffe *false positiv* und *false negativ* für den falsch klassifizierten Anteil stehen.

2.4.1 Präzision

Die Präzision, auch User Accuracy (UA) genannt, gibt an, welcher Anteil des von einem Modell klassifizierten Datensatzes richtig zugeordnet wurde. Der Wertebereich der Präzision liegt zwischen 0 und 1. Die Präzision wird wie folgt berechnet:

$$\mathrm{UA} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.15}$$

2.4.2 Trefferquote

Die Trefferquote, auch Producer Accuracy gennant (PA), repräsentiert den von einem Modell richtig klassifizierten Anteil im Verhältnis zum gesamten Datensatz. Der Wertebereich der Trefferquote liegt zwischen 0 und 1. Die Trefferquote wird wie folgt berechnet:

$$\mathrm{PA} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.16}$$

2.4.3 Gesamtgenauigkeit

Die Gesamtgenauigkeit, auch Total Accuracy (TA) genannt, beschreibt die Genauigkeit eines Modells im Bezug auf alle Klassen. Es gibt an, wie groß das Verhältnis des richtig klassifizierten Anteils aller Klassen zum gesamten Datensatz ist. Die Verlässlichkeit dieser Metrik ist sehr von der Verteilung der Klassen abhängig. Eine verlässliche Metrik kann durch eine gleiche Verteilung aller Klassen erreicht werden. Die Gesamtgenauigkeit kann Werte zwischen 0 und 1 annehmen und wird nach folgender Formel berechnet:

$$\mathrm{TA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2.17}$$

2.4.4 F_1-Maß

Das F_1-Maß bietet die Möglichkeit, sowohl die Präzision als auch die Trefferquote gleichermaßen bei der Bewertung des Klassifikationsergebnisses zu berücksichtigen (Taha und Hanbury, 2015). Der Wertebereich des F_1-Maßes liegt zwischen 0 und 1. Das F_1-Maß wird wie folgt berechnet:

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{UA} \cdot \mathrm{PA}}{\mathrm{UA} + \mathrm{PA}} \tag{2.18}$$

2.4.5 Jaccard-Koeffizient

Ob eine Objektdetektion erfolgreich war, wird anhand einer speziellen Metrik bestimmt. Bei dieser Metrik handelt es sich um den Jaccard-Koeffizienten, der auch Intersection over Union (IoU) genannt wird. Mit diesem Koeffizienten wird die Ähnlichkeit der Objektpaare ermittelt. Im Fall der Objektdetektion besteht ein Objektpaar aus der vordefinierten Bounding Box eines Objektes und der vom Netz vorhergesagten Bounding Box. Der IoU-Koeffizient kann Werte zwischen 0 und 1 annehmen, wobei 0 keine Ähnlichkeit bedeutet und 1 für eine komplette Übereinstimmung steht (Backhaus u. a., 2015). Der IoU-Koeffizient für zwei Mengen A und B wird wie folgt berechnet:

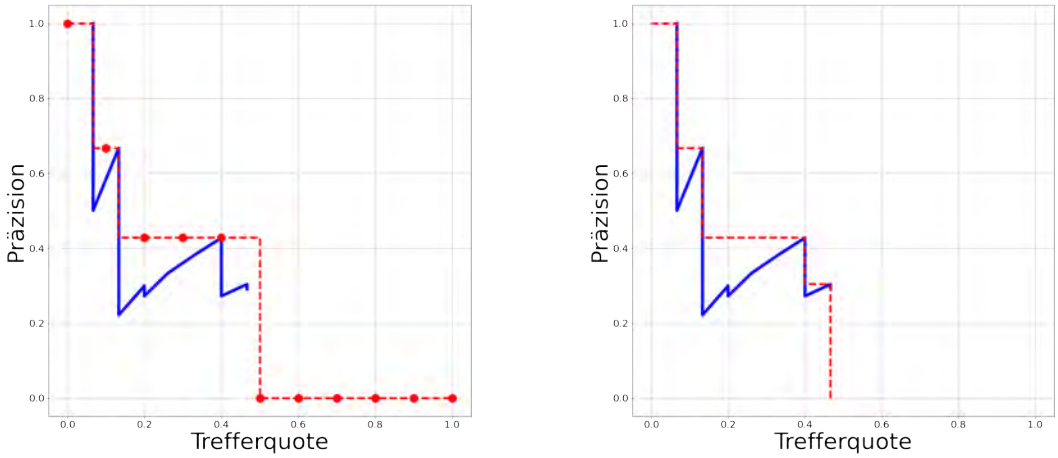
$$\mathrm{IoU} = \frac{\mathrm{A} \cap \mathrm{B}}{\mathrm{A} \cup \mathrm{B}} \tag{2.19}$$

Für die Bewertung der Ergebnisse einer Objektdetektion wird ein IoU-Grenzwert festgelegt. Ein IoU-Grenzwert von z.B. 0,5 bedeutet, dass die Übereinstimmung zwischen einer vorhergesagten Bounding Box mit einer vordefinierten Bounding Box mindestens 50% betragen muss, damit die Detektion als richtig eingestuft wird.

2.4.6 Mittlere durchschnittliche Präzision

Eine einheitliche Leistungsmetrik für die Bewertung der Objektdetektion ist die mittlere durchschnittliche Präzision (Padilla u. a., 2020), auf Englisch mean Average Precision (mAP) (vgl. 2.20). Die mAP ist eine über N Klassen gemittelte durchschnittliche Präzision (AP) (vgl. 2.21). Der Wertebereich der mAP liegt auch wie bei der Präzision und der Trefferquote zwischen 0 und 1.

$$\text{mAP} = \frac{1}{\text{N}} \sum_{i=1}^{N} \text{AP}_i \tag{2.20}$$



(a) Verlauf der sogenannten 11-Point-Interpolation.

(b) Verlauf der sogenannten All-Point-Interpolation.

Abbildung 2.11: Ein Vergleich zweier unterschiedlicher Näherungsverfahren für die Berechnung der durchschnittlichen Präzision (AP). Die blaue Kurve zeigt den tatsächlichen Verlauf der Präzision-Trefferquote-Kurve für 24 Messungen. Mit der roten Linie ist jeweils das Ergebnis eines speziellen Interpolationsverfahrens dargestellt, so wie es in Padilla u. a. 2020 beschrieben ist (Quelle: Padilla u. a. 2020, modifiziert).

Die durchschnittliche Präzision (AP) wird aus dem Integral über die Präzision in Abhängigkeit von der Trefferquote berechnet. In der Praxis wird das Integral durch eine Näherung ersetzt. Dazu wird der Verlauf der Präzision-Trefferquote-Kurve nach einem Verfahren gemäß 2.21 und 2.22 vereinfacht. Für dieses spezielle Interpolationsverfahren werden entweder n Punkte oder alle Punkte der Kurve verwendet, wobei n für eine vordefinierte Anzahl der Punkte steht (vgl. Abbildung 2.11). Um eine Vergleichbarkeit von den Detektionsergebnissen der verschiedenen Algorithmen zu erreichen, muss eine einheitliche Anzahl der Interpolationspunkte für die Berechnung der AP verwendet werden. Für die Teilnahme an dem Pascal Visual Object Classes Wettbewerb (Everingham u. a., 2015) und an dem COCO Detektion Wettbewerb (COCO, 2019) sind z.B. 11 bzw. 101 Interpolationspunkte vorgegeben (Padilla u. a., 2021).

Für den Vergleich der Detektionsergebnisse in diesem Projekt wird der Kurvenverlauf über alle Punkte interpoliert (vgl. 2.21 und 2.22).

$$\mathrm{AP} = \sum_{n} (\mathrm{PA}_{n+1} - \mathrm{PA}_{n}) \mathrm{UA}_{interp}(\mathrm{PA}_{n+1}) \ , \tag{2.21}$$

wobei

$$\mathrm{UA}_{interp}(\mathrm{PA}_{n+1}) = max\mathrm{UA}(\widehat{\mathrm{PA}}), \text{ mit } \widehat{\mathrm{PA}} \geq \mathrm{PA}_{n+1} \tag{2.22}$$

2.4.7 Kappa-Koeffizient

Ein weiteres Maß für die Beurteilung der Klassifikaionsergebnisse ist der Kappa-Koeffizient (Cohen, 1960). Ähnlich wie das F_1-Maß wird die Präzision und die Trefferquote indirekt bei der Berechnung des Kappa-Koeffizients berücksichtigt. Somit kann ein zufälliges Klassifikationsergebnis eliminiert werden. Der Kappa-Koeffizient kann Werte zwischen -1 und 1 annehmen, wobei positive Werte zu erwarten sind (Congalton, 2001). Werte kleiner als 0,4 deuten auf ein schlechtes Klassifikationsergebnis hin (De Lange, 2013). Der Kappa-Koeffizient kann mit folgender Formel beschrieben werden (Taha und Hanbury, 2015):

$$\kappa = \frac{\mathrm{f}_a - \mathrm{f}_c}{\mathrm{N} - \mathrm{f}_c} \tag{2.23}$$

Dabei gibt N die Größe des untersuchten Datensatzes an. f_a und f_c werden nach den Formeln 2.24 und 2.25 berechnet.

$$\mathrm{f}_a = \mathrm{TP} + \mathrm{TN} \tag{2.24}$$

$$\mathrm{f}_c = \frac{(\mathrm{TN} + \mathrm{FN}) \cdot (\mathrm{TN} + \mathrm{FP}) + (\mathrm{FP} + \mathrm{TP}) \cdot (\mathrm{FN} + \mathrm{TP})}{\mathrm{N}} \tag{2.25}$$

3 Daten

In diesem Kapitel wird beschrieben, welche Fernerkundungsdaten in diesem Projekt verwendet werden und aus welchen Untersuchungsgebieten diese Daten stammen.

3.1 Untersuchungsgebiete

In diesem Projekt werden Fernerkundungsdaten von insgesamt acht Landkreisen in Bayern verwendet (Abb. 3.1, Tab. 3.1).

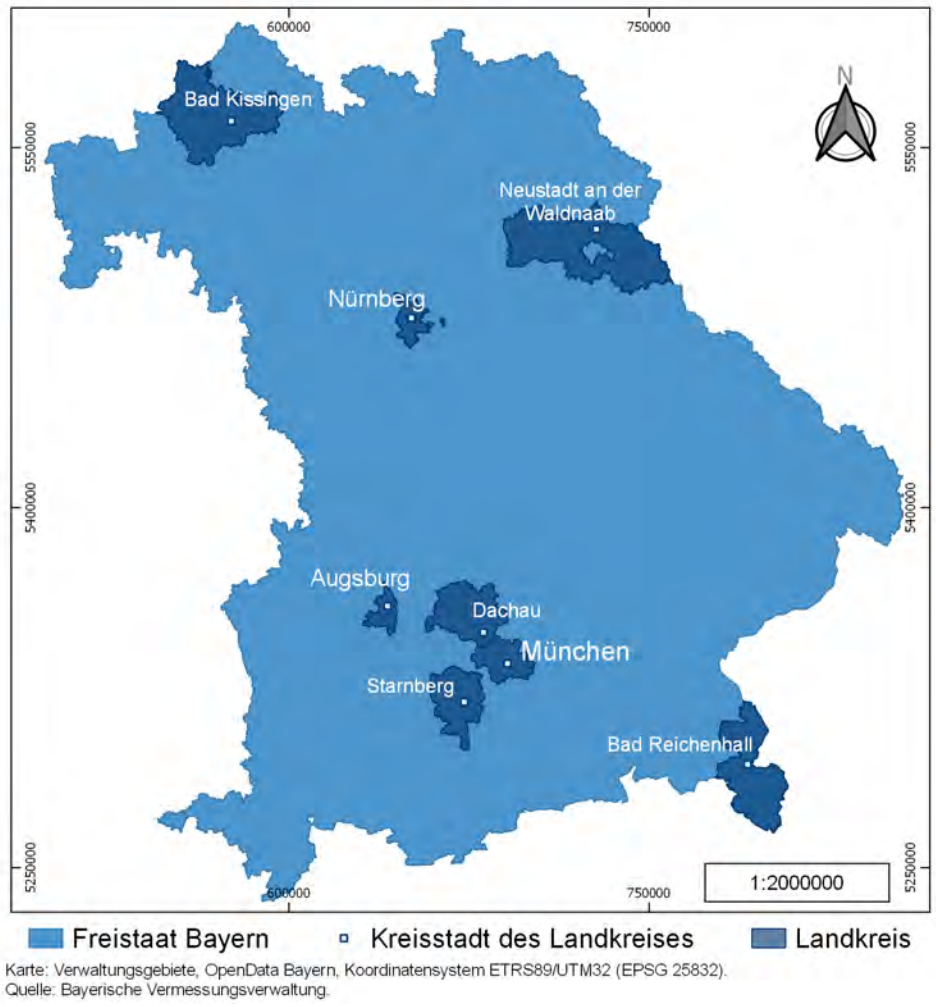


Abbildung 3.1: Untersuchte Landkreise im Freistaat Bayern.

Der größte Teil der Trainings- und Validierungsdaten für die neuronalen Netze YOLOv3 und RetinaNet stammen aus dem Gebiet der Stadt München. Die Verfügbarkeit von Referenzdaten mit bekannten Baumstandorten spielt bei der Auswahl der Trainingsgebiete eine entscheidende Rolle. Die Referenzdaten aus der Stadt München wurden durch eine Straßenbefahrung gewonnen und repräsentieren für ein städtisches Gebiet typische Baumarten wie Linden, Ahorn, Eiche, Platanen und Rosskastanien (SWR, 2022). Auch die Lage des Untersuchungsgebietes in der Nähe von München ist günstig, da diese eine Begehung des Gebietes zwecks einer Vor-Ort-Verifizierung der Detektionsergebnisse vereinfacht. So können zum Beispiel die Gründe einer Fehldetektion untersucht werden.
Weitere Referenzdaten werden aus dem Baumkataster des Landkreises Dachau und aus einer Vermessungskampagne des Landkreises Berchtesgadener Land entnommen. Bei diesem Teil der Daten handelt es sich hauptsächlich um Bäume aus Wäldern und Parks, wodurch die Diversität der Baumarten im ganzen Datensatz erhöht wird. Diese Daten entsprechen nur einem kleinen Anteil aller Referenzdaten, da die Aufbereitung der Daten aus den Gebieten mit einer dichten Bewaldung mit einem sehr hohen Arbeitsaufwand verbunden ist.
Für die unabhängige Auswertung der Detektionsergebnisse des vortrainierten neuronalen Netzes werden die Gebiete aus den Landkreisen Starnberg, Neustadt an der Waldnaab und Bad Kissingen sowie aus den Städten München, Augsburg und Nürnberg verwendet. Darunter sind sowohl Stadtbereiche, als auch Randgebiete von kleinen Siedlungen enthalten. Die Daten der ausgewählten Gebiete wurden in unterschiedlichen Zeiträumen und von unterschiedlichen Befliegungsfirmen aufgenommen. Diese Faktoren werden berücksichtigt, um das Detektionsergebnis in Bezug auf mögliche Abweichungen in der Radiometrie der Luftbilder zu untersuchen.

Tabelle 3.1: Liste der untersuchten Gebiete und wie sie im Projekt verwendet werden. Bei der Anwendung wird zwischen Objektdetektion (OD) und Auswertung (A) der Detektionsergebnisse unabhängig vom Testdatensatz unterschieden.

Gebiet	**Befliegungstag**	**Anwendung**
Stadt Augsburg	22. April 2020	A
Stadt München	26. April 2020	OD, A
Stadt Nürnberg	01. Juni / 03. Juni 2021	A
Landkreis Bad Kissingen, Stadt Bad Kissingen	03. September 2021	A
Landkreis Berchtesgadener Land, Gemeinde Schönau am Königssee	21. August 2020	OD
Landkreis Dachau, Stadt Dachau	25. April / 26. April 2020	OD
Landkreis Neustadt an der Waldnaab, Stadt Windischeschenbach	09. Juni 2019	A
Landkreis Starnberg, Ortsteil Stockdorf	26. April 2020	A

3.2 Datengrundlage

Als Hauptdatengrundlage dienen Digitale Oberflächenmodelle, Digitale True Orthophotos und Digitale Geländemodelle der Untersuchungsgebiete, die vom Landesamt für Digitalisierung, Breitband und Vermessung Bayern (LDBV) zu Verfügung gestellt wurden.
Alle Daten liegen in Form einzelner Kacheln im Koordinatensystem ETRS89 mit der Zone UTM

32 (EPSG 25832) vor. Jede Kachel entspricht einer Fläche von 1 km^2.
Die digitalen Luftbilder, die für die Erstellung der Digitalen Oberflächenmodelle und der Digitalen True Orthophotos verwendet werden, wurden im Rahmen der Bayernbefliegung aufgenommen. Die Luftbildaufnahme erfolgte senkrecht zur Erdoberfläche aus einer Höhe von ca. 4 km.
Für die Aufnahme wurden digitale Großformat-Kameras verwendet, deren lichtempfindliche Sensoren das reflektierte Sonnenlicht im Bereich von 400 nm bis 1100 nm erfassen. Dieser Wellenlängenbereich beinhaltet sowohl sichtbares Licht als auch nahes Infrarot (NIR). Zu dem sichtbaren Licht gehören blaue, grüne und rote Anteile des Lichtspektrums, deren mittlere Wellenlänge 435 nm, 546 nm, und 700 nm entspricht (Süße und Rodner, 2014). Die Wellenlängen von 800 nm bis 1100 nm werden als das Nahinfrarot bezeichnet (De Lange, 2013).
Seit dem Jahr 2017 erfolgt jede Bayernbefliegung mit einer Längsüberdeckung von 80% und einer Querüberdeckung von 50%. Diese hohen Bildüberlappungen ermöglichen eine dreidimensionale Auswertung der Aufnahmen.
Sowohl Digitale Oberflächenmodelle als auch Digitale True Orthophotos wurden mit einer Bodenpixelgröße von 20 cm ausgeliefert.
Die 3D-Punktwolken, die als Datengrundlage für die Erstellung der Digitalen Geländemodelle dienen, wurden mittels flugzeuggestütztem Laserscanning erfasst. Seit dem Jahr 2011 haben alle Punktwolken eine Punktdichte von mindestens 4 Punkte/m^2.
Die Digitalen Geländemodelle stehen in Form von ASCII-Dateien zur Verfügung. Diese Dateien enthalten die georeferenzierten X-, Y-, und Z-Koordinaten der einzelnen Punkte einer Punktwolke.

3.2.1 Digitales Oberflächenmodell (DOM)

Ein bildbasiertes Digitales Oberflächenmodell beschreibt die Erdoberfläche inklusive der Objekte, die sich darauf befinden. Dazu zählen z.B. Bauwerke und Vegetation. Ein DOM wird aus den orientierten Stereoluftbildpaaren gewonnen.
Für die Orientierung eines überlappenden Bildpaares müssen die Parameter der inneren und äußeren Orientierung bekannt sein. Zu der inneren Orientierung einer Kamera zählen die Koordinaten des Bildhauptpunktes und die Brennweite der Kamera. Die äußere Orientierung schließt die Koordinaten des Projektionszentrums und drei Drehwinkel ein.
Eine Orientierung der Luftbilder erfolgt in der Aerotriangulation. In der anschließenden dichten Bildzuordnung werden aus den Stereobildpaaren homologe Punkte extrahiert und aus jedem Paar homologer Punkte ein 3D-Punkt berechnet. Die berechneten 3D-Punkte bilden eine erste 3D-Rekonstruktion der aufgenommenen Szene mit einer dünnen Punktwolke. Die Rekonstruktion in Form einer dichten Punktwolke wird mit dem Semi-global Matching Verfahren durchgeführt.
Für die Erstellung eines DOM wird aus der dichten Punktwolke ein regelmäßiges Gitter modelliert. Dabei wird nur ein Abstandswert pro Gitterzelle berechnet. Somit beinhaltet ein bildbasiertes DOM keine 3D-Information mehr, sondern nur die 2,5D-Information.

3.2.2 Digitales True Orthophoto (TrueDOP)

Die Digitalen Orthophotos, die im Rahmen dieses Projektes verwendet werden, entsprechen einer Parallelprojektion der aufgenommenen Szene. Somit werden die Eigenschaften eines True Orthophotos erfüllt. Die in einem TrueDOP abgebildeten Objekte sind maßstabsgetreu und weisen keine Kippeffekte am Rand des Bildes auf. So werden sichttote Räume in Bildern vermieden.

Für jeden Höhenwert im DOM werden aus den am besten passenden Luftbildern Farbwerte der vier Spektralkanäle ermittelt. Somit entsteht ein multispektrales TrueDOP. Im Unterschied zu einem DOM enthält ein TrueDOP nur 2D-Information.
Die multispektrale Information beschreibt die Reflektivität der Objekte auf der Erdoberfläche. Die Reflektivität berechnet sich aus dem Verhältnis der reflektierten elektromagnetischen Sonnenstrahlung zur einfallenden Sonnenstrahlung. Dabei wird die Intensität der Reflektivität für bestimmte Wellenlängen der Strahlung erfasst.
Die Art, wie Objekte die Sonnenstrahlung reflektieren, ist von den physikalischen Eigenschaften der Objekte abhängig (vgl. Abbildung 3.2). Die multispektrale Information ist die wichtigste Datengrundlage in diesem Projekt, da sie es ermöglicht, die aufgenommenen Objekte durch ihre Reflektivität zu unterscheiden, wodurch wiederum eine Klassifikation mit den Methoden des maschinellen Lernens möglich ist.
Alle True Orthophotos für dieses Projekt liegen mit einer 8-Bit-Farbtiefe je Kanal vor.

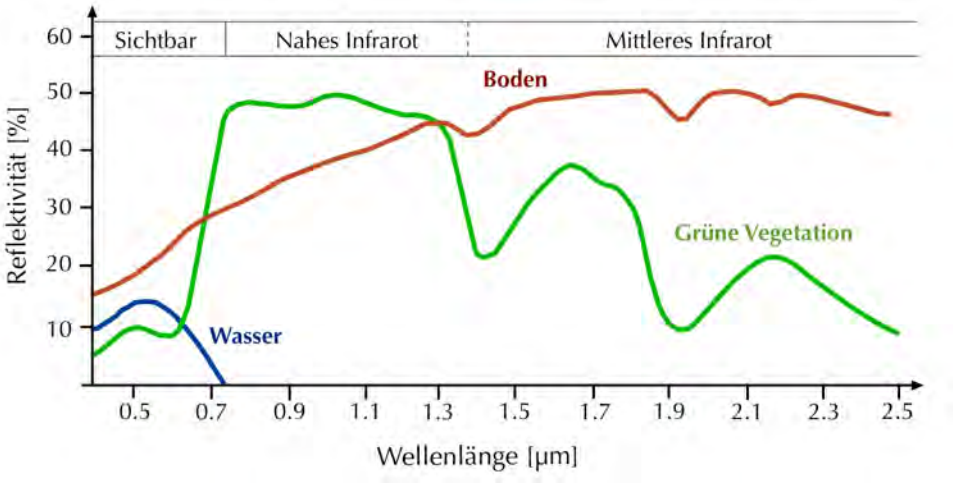


Abbildung 3.2: Das Reflexionsverhalten ausgewählter Objektoberflächen (Quelle: SEOS 2012).

3.2.3 Digitales Geländemodell (DGM)

Ein Digitales Geländemodell beschreibt den Verlauf des Geländes ohne die Objekte wie Bauwerke oder Vegetation auf der Erdoberfläche. Für die Erstellung eines DGM wird eine 3D-Punktwolke mittels flugzeuggestütztem Laserscanning erfasst.
Die Erfassung erfolgt mit einem Full Waveform Laserscanner. Die Befliegung für die Erfassung der Laserpunktwolke wurde in der vegetationsarmen Zeit durchgeführt, um die Anzahl der Reflektionen vom Boden zu erhöhen. Aus der erfassten Punktwolke werden Bodenpunkte durch eine automatische Filtermethode extrahiert und anschließend zu einem regelmäßigen Gitter interpoliert. Die Bodenpixelgröße des Gitters beträgt 1 m.

3.2.4 Normalisiertes Digitales Oberflächenmodell (nDOM)

Ein normalisiertes Digitales Oberflächenmodell bildet die relative Höhe von Objekten auf der Erdoberfläche über dem Gelände ab. Die Grundlage für die Erstellung eines nDOM ist ein

Digitales Oberflächenmodel und ein Digitales Geländemodell. Die Berechnung des nDOM erfolgt nach folgender Formel:

$$\text{nDOM} = \text{DOM} - \text{DGM} \tag{3.1}$$

Da die geometrische Auflösung des DGM nicht mit der geometrischen Auflösung des DOM übereinstimmt, wird vor der Berechnung des nDOM eine Interpolation des DGM durchgeführt. Dabei wird das regelmäßige Gitter des DGM auf eine Bodenpixelgröße von 20 cm hochgerechnet. Somit wird erreicht, dass die Bodenpixelgröße des nDOM und des TrueDOP einheitlich ist. Dies ermöglicht die Fusion von nDOM und TrueDOP im Rasterformat. Das ist für dieses Projekt von besonderer Bedeutung, damit die Vegetation mit einer bestimmten Höhe klassifiziert werden kann.

3.2.5 Normalisierter Vegetationsindex (NDVI)

NDVI steht für Normalized Difference Vegetation Index. Mit diesem Index wird das Verhältnis zwischen den Reflektivitäten im roten und im nahinfraroten Lichtspektrum beschrieben (Weier und Herring, 2000). Die Berechnung erfolgt mit den Spektralwerten der TrueDOP. Der NDVI wird nach folgender Formel berechnet:

$$\text{NDVI} = \frac{\text{NIR} - \text{Rot}}{\text{NIR} + \text{Rot}} \tag{3.2}$$

Der NIR-Kanal, der für diese Berechnung verwendet wird, ist besonders wichtig für die Erkennung der Vegetation allgemein und explizit für die Unterscheidung verschiedener Vegetationsarten untereinander (EOS Data Analytics, 2022). Wie in Abbildung 3.2 zu sehen, reflektieren die Pflanzen diesen bestimmten Wellenlängenbereich des Lichtspektrums, im Gegensatz zu anderen Objekten, stärker.
Der NDVI kann Werte zwischen -1 und 1 annehmen. Für die Vegetation sind Werte im positiven Bereich zu erwarten (Roettger, 2007). Dabei ist zu erwähnen, dass die Reflektivitäten der Vegetation stark von der Jahreszeit abhängig sind. Damit sind Veränderungen des NDVI für die Vegetation im Laufe des Jahr verbunden.
Die NDVI Werte sind eine wichtige Datengrundlage bei der Vegetationsklassifikation. Dadurch werden die unterschiedlichen Reflektivitäten der Vegetation und der Nichtvegetation hervorgehoben.

4 Werkzeuge

In diesem Kapitel werden die wichtigsten Werkzeuge vorgestellt, die für die Realisierung dieses Projektes eingesetzt werden.

4.1 Software

In folgenden Unterkapiteln wird die Software näher beschrieben, welche die Aufbereitung der Daten sowie die Verwendung der Algorithmen des maschinellen Lernens in diesem Projekt ermöglicht.

4.1.1 QGIS

QGIS ist ein Geoinformationssystem, mit dem räumliche Informationen im Raster- und Vektorformat analysiert und bearbeitet werden können (QGIS, 2002).
Es ist eine freie Open Source Software, die unter der General Public License (GPL) betrieben wird. Das bedeutet, dass die Benutzer die Software frei verwenden, untersuchen, anpassen und verbreiten dürfen. Die QGIS-Architektur bietet Programmierschnittstellen, um auch externe Erweiterungen in das Programm leicht integrieren zu können.
QGIS ist eine betriebssystemübergreifende Software und unterstützt die Betriebsysteme Windows, Linux, Android und MacOS.
In diesem Projekt wird mit der QGIS-Version 3.4.5 gearbeitet. Eine wichtige Rolle werden die QGIS-Funktionen spielen, die für die Aufbereitung der Daten im Vorverarbeitungsschritt verwendet werden. Zu nennen wäre da die Funktion Rasterrechner, die die Berechnung auf den Rasterpixelwerten ermöglicht. Außerdem wird QGIS für die visuelle Auswertung der Endergebnisse eingesetzt.

4.1.2 GDAL

GDAL steht für Geospatial Data Abstraction Library (GDAL, 1998). Es ist eine Sammlung freier Open Source Funktionen, die Analyse- und Bearbeitungswerkzeuge für räumliche Raster- und Vektordaten implementieren. Nach der Installation lassen sich die Funktionen der GDAL sowohl aus einem Programmcode abrufen als auch in ein vorhandenes Programm mit einer grafischen Benutzeroberfläche wie QGIS integrieren. Solche Programmierschnittstellen für die GDAL werden in den meisten Geoinformationssystemen unterstützt.
Die GDAL wird sowohl in Form einer in QGIS integrierten Erweiterung als auch direkt aus einem Python-Skript in diesem Projekt verwendet. Mit der GDAL-Funktionen werden die gewünschten Spektralkanalkombinationen der Rasterdaten erstellt, Rasterdaten auf eine Vektordatenmaske verschnitten und Rasterpixelwerte für eine weitere Auswertung extrahiert.

4.1.3 Python

Python ist eine objektorientierte Programmiersprache, die im Jahr 1991 entwickelt wurde (Python, 1991). Dank der relativ einfachen Syntax im Vergleich zu anderen höheren Programmiersprachen gewann Python im Laufe der Zeit an Bedeutung. Seit den letzten Jahren hat sich die Sprache im Bereich der Datenanalyse und des maschinellen Lernens etabliert. Die aktuelle Version der Sprache wurde im Jahr 2008 vorgestellt und läuft unter der Bezeichnung Python 3.
Die hohe Anzahl an frei verfügbaren Software-Bibliotheken und Implementierungen ist der Grund dafür, dass diese Programmiersprache für das Projekt ausgewählt wird.
Für die Installation der Python-Bibliotheken wird im Rahmen dieses Projektes die Anaconda-Umgebungen verwendet. Anaconda bietet eine unkomplizierte Lösung für die Verwaltung von Python-Bibliotheken, wodurch mögliche Konflikte zwischen unterschiedlichen Software-Versionen vermieden werden (Anaconda, 2009).

4.1.4 PyTorch

PyTorch ist eine freie Open Source Programmbibliothek, die von dem Unternehmen Meta Platforms (ehemals Facebook) entwickelt wurde (PyTorch, 2016). Die Bibliothek bietet zahlreiche Werkzeuge für die Realisierung von Projekten im Bereich des maschinellen Lernens. Vor allem ist die Bibliothek für die Flexibilität beim Erstellen, Trainieren und Betreiben von neuronalen Netzen bekannt. Das Konzept von PyTorch vereint die Ideen von zwei früheren Bibliotheken Torch und Chainer. Laut diesem Konzept wird der Entwicklungsprozess von neuronalen Netzen dynamisch gestaltet. Somit kann der Entwickler mit den einzelnen Bausteinen eines neuronalen Netzes (im Weiteren auch das Modell genannt) experimentieren, ohne das Modell als Ganzes jedes mal neu kompilieren zu müssen.
Dank einer integrierten Lösung für die Programmierschnittstelle CUDA können die Berechnungen der PyTorch-Funktionen durch einen Nvidia Grafikprozessor (GPU) ausgeführt werden. Die GPU ermöglicht eine parallele Verarbeitung großer Datenmengen, die für das Trainieren von neuronalen Netzen erforderlich sind. Durch diese Parallelisierung werden die Berechnungen erheblich beschleunigt (Nvidia Developer, 2022).
Die neuronalen Netze, die in diesem Projekt eingesetzt werden, wurden mittels PyTorch-Bibliothek implementiert.

4.1.5 Scikit-learn

Scikit-learn ist eine freie Software-Bibliothek mit effizienten Implementierungen der wichtigsten Methoden des maschinellen Lernens (Scikit-learn, 2010). Die umfassende Online-Dokumentation wird von den Entwicklern der Bibliothek bereitgestellt. Dies ermöglicht einen einfachen Einstieg und die Verwendung der Bibliothek-Funktionen.
In diesem Projekt wird die Implementierung des K-Nearest-Neighbors-Algorithmus aus der Scikit-learn-Bibliothek verwendet.

4.1.6 LabelImg

LabelImg ist eine Software für die Erstellung gelabelter Datensätze, die im Bereich der Objektdetektion mit neuronalen Netzen verwendet werden können (LabelImg, 2015). Die grafische Benutzeroberfläche der Software ermöglicht eine einfache Erstellung der Label. Dabei werden einzelne Objekte direkt in den Bildern markiert. Das Format, in dem die Label gespeichert werden, kann vom Benutzer eingestellt werden. Zur Zeit werden Datenformate wie PASCAL VOC XML, CreateML JSON und YOLO TXT unterstützt. LabelImg ist eine freie Software, die in Python geschrieben wurde.
Alle Datensätze für die Objektdetektion im diesem Projekt werden mit LabelImg erstellt.

4.2 Hardware

Die Arbeitsstation, die für alle Experimente verwendet wird, verfügt über zwei Hauptprozessoren AMD Epyc 7302 mit 16 Kernen und einem Gesamtarbeitsspeicher von 512 GB. Eine weitere Recheneinheit ist der Grafikprozessor Nvidia Quadro RTX 8000 mit einer Speicherkapazität von 48 GB. Das Gesamtsystem läuft unter dem Betriebsystem Windows Server 2019 Standard.

5 Methodik

In diesem Kapitel wird der allgemeine Arbeitsablauf und die Vorgehensweise bei der Vorbereitung der einzelnen Experimente beschrieben (vgl. Abbildung 5.1). Dabei wird kurz auf die theoretischen Hintergründe der Funktionsweise der verwendeten Algorithmen eingegangen.

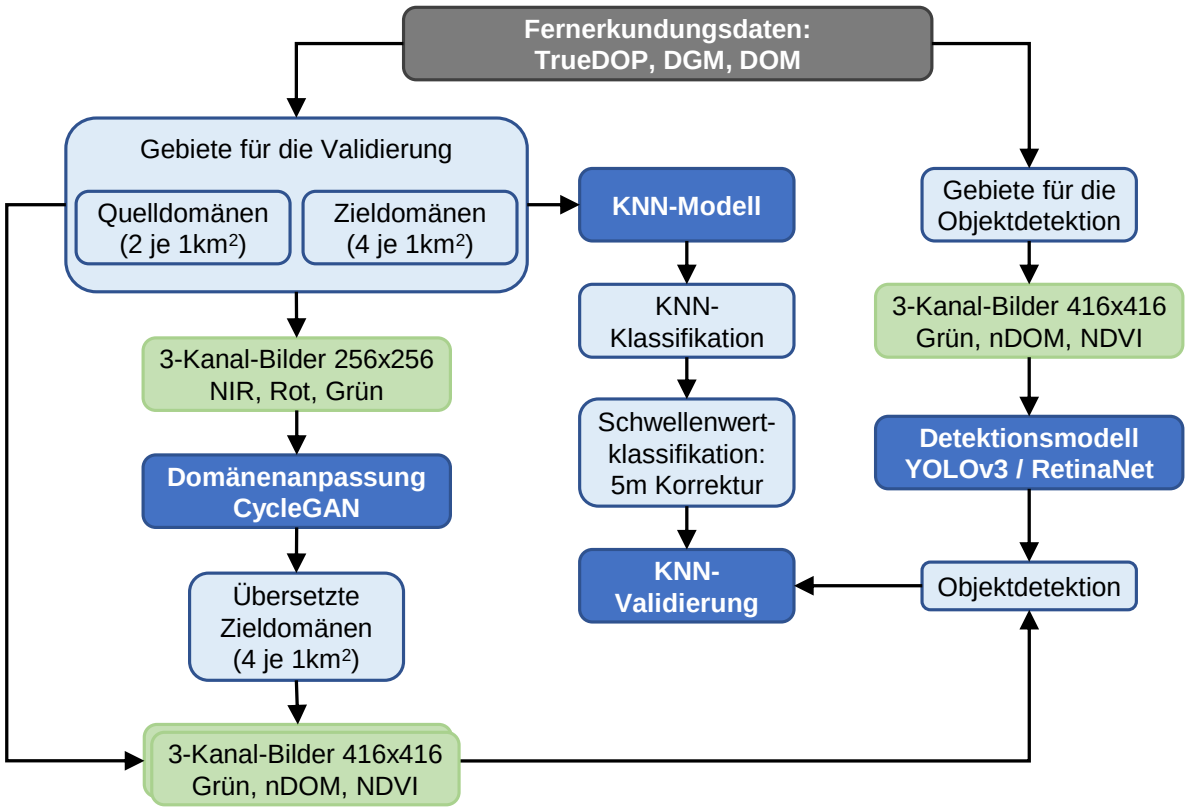


Abbildung 5.1: Der allgemeine Ablauf der wichtigsten Experimente des Projekts.

Im ersten Schritt werden die neuronalen Netze YOLOv3 und RetinaNet für die Einzelbaumdetektion trainiert. Dafür wird der Trainings-, der Validierungs- und der Testdatensatz erstellt. Für die Erstellung dieser Datensätze werden Fernerkundungsdaten aus dem Gebiet der Stadt München sowie aus dem Landkreis Dachau und dem Landkreis Berchtesgadener Land verwendet. Das Detektionsergebnis beider Netze wird auf dem Testdatensatz mit Labels ausgewertet.
Im nächsten Schritt wird das Detektionsmodell mit dem besten Ergebnis auf sechs verschiedene Gebieten ohne Labels angewendet. Dabei handelt es sich jeweils um ein $1\,\mathrm{km}^2$ großes Gebiet im Ortsteil Stockdorf und in den Städten Windischeschenbach, Bad Kissingen, München, Augsburg und Nürnberg.
Die Verteilung der spektralen Information des Rot-, des Grün-, des Blau- und des Nahinfrarotkanals der TrueDOPs der Gebiete in Windischeschenbach, Bad Kissingen, Augsburg und

Nürnberg unterscheidet sich von der Verteilung der spektralen Information der TrueDOPs aus dem Gebiet der Stadt München, mit denen das Detektionsmodell trainiert wird. Solche Abweichungen können das Detektionsergebnis verschlechtern. Um den Einfluss dieser Abweichungen zu untersuchen, wird eine Domänenanpassung durchgeführt. Die Übersetzung der Bilder zwischen der Ziel- und der Quelldomäne wird mit dem neuronalen Netz CycleGAN durchgeführt. Für die Übersetzung der städtischen Gebiete der Städte Augsburg und Nürnberg werden die Daten aus der Stadt München als Quelldomäne verwendet. Die Daten aus dem Ortsteil Stockdorf dienen der Übersetzung der ländlichen Gebiete der Städte Bad Kissingen und Windischeschenbach. Dies ist wichtig, um eine allgemeine Ähnlichkeit der Verteilungen der Ziel- und der Quelldomäne zu gewährleisten. Nach der Domänenanpassung wird das Detektionsmodell auch auf die übersetzten Bilder angewendet.
Anschließend wird die Auswertung der Detektionsergebnisse unabhängig vom Testdatensatz durchgeführt. Dafür werden die oben genannten Untersuchungsgebiete zuerst mittels K-Nearest-Neighbors-Algorithmus klassifiziert. Danach wird die Zusammensetzung der detektierten Bounding Boxen pixelweise mit dem Klassifikationsergebnis verglichen. Die Bounding Boxen, welche laut KNN-Klassifikation keine Vegetationspixel enthalten, werden als Fehldetektion betrachtet. So wird eine Fehlerquote für die einzelne Gebiete bestimmt. Außerdem wird bei dieser Untersuchung der Anteil der detektierten Vegetation an der klassifizierten Gesamtvegetation ermittelt.
Sowohl die Fehlerquote als auch der Anteil der detektierten Vegetation an der Gesamtvegetation werden vor und nach der Domänenanpassung ausgewertet.

5.1 Einzelbaumdetektion

In diesem Projekt werden die Netze YOLOv3 und RetinaNet für die Einzelbaumdetektion eingesetzt. Diese werden in den kommenden Unterkapiteln näher beschrieben. Dabei wird auch angesprochen, wie die Datensätze für das Trainieren der Netze erstellt werden und welche Einstellungen während des Trainings verwendet werden.

5.1.1 YOLOv3

Das Akronym YOLO steht für You Only Look Once. Den Namen des Netzes erklären die Autoren damit, dass jedes Bild nur einmal durch das Netz bearbeitet wird, um Objekte zu lokalisieren und zu klassifizieren. Für dieses Projekt wird die Version 3 des Netzes verwendet, die im Jahr 2018 vorgestellt wurde (Redmon und Farhadi, 2018).
Das Gerüst von YOLOv3 bildet das CNN namens Darknet-53. Das CNN besteht aus 53 Faltungsschichten. In YOLOv3 wurde das Darknet-53 durch weitere 53 Schichten ergänzt, sodass das Netz insgesamt 106 Schichten beinhaltet. Die Schichten sind mit der Aktivierungsfunktion Leaky ReLU verbunden. Der Fallkoeffizient a ist auf 0,1 gesetzt (vgl. Abschnitt 2.1.1).
Für die Objektdetektion werden die Merkmalskarten aus den Schichten 82, 94 und 106 verwendet. Die Größe der Merkmalskarten in diesen Schichten beträgt jeweils $1/32$, $1/16$ und $1/8$ der Eingabebildgröße. Somit werden die Objekte auf drei unterschiedlichen Generalisierungsstufen detektiert. Die Objektdetektion in unterschiedlichen Generalisierungsstufen wurde von Lin u. a. (2017a) vorgestellt. Dieses Konzept hat sich in verschiedenen Detektionsalgorithmen etabliert.
Für jedes Pixel der Merkmalskarte wird in jeder Generalisierungsstufe ein Vektor vorhersagt (vgl. Abbildung 5.2). Dieser Vektor wird aus P Elementen zusammengesetzt. Die Anzahl der Elemente P hängt von der Anzahl der Klassen n im Trainingsdatensatz und der Anzahl der

Objektrahmen B ab, welche für dieses Pixel erstellt werden.
In YOLOv3 werden pro Merkmalskartenpixel drei Objektrahmen mit unterschiedlichen Größen erstellt. Die Berechnung der Objektrahmen erfolgt in mehreren Schritten. Vor Beginn des Trainings werden für alle drei Detektionsschichten sogenannte Anker-Objektrahmen in unterschiedlichen Größen vordefiniert. Die Größe der Anker-Objektrahmen wird mittels K-Means-Clusteranalyse bestimmt. Mit diesem Verfahren wird anhand der vordefinierten Objektrahmen im Trainingsdatensatz die optimale Größe für die Anker-Objektrahmen geschätzt. Diese Methode für die Bestimmung der Größen für die Anker-Objektrahmen zeigt eine positive Wirkung auf das Lernen des Netzes. Als Alternative können auch die Anker-Objektrahmen verwendet werden, die für einen anderen Datensatz berechnet wurden. In der Implemetierung des Netzes YOLOv3, die in diesem Projekt verwendet wird, werden die Anker-Objektrahmen verwendet, welche auf dem Datensatz COCO angepasst wurden.
Die tatsächliche Größe eines Objektrahmens muss durch die Anpassung der Anker-Objektrahmen berechnet werden. Wie die Anker-Objektrahmen angepasst werden müssen, ist ein Teil des Lernens des Netzes.
Für jeden Objektrahmen werden vier Parameter von dem Netz vorhergesagt, welche die Höhe $\hat{h}$ und die Breite $\hat{w}$ des Objektrahmens sowie die $\hat{c}_x$- und $\hat{c}_y$-Koordinaten des Rahmenzentrums definieren. Diese vier Parameter entsprechen den ersten vier Elementen des Vektors.
Ein weiteres Element ist die vorhergesagte Wahrscheinlichkeit $\hat{C}$, dass dieser Pixel ein Objekt enthält. Dieser Wert liegt zwischen 0 und 1, wobei 1 für das Zentrum eines Objekts und 0 für kein Objekt steht.

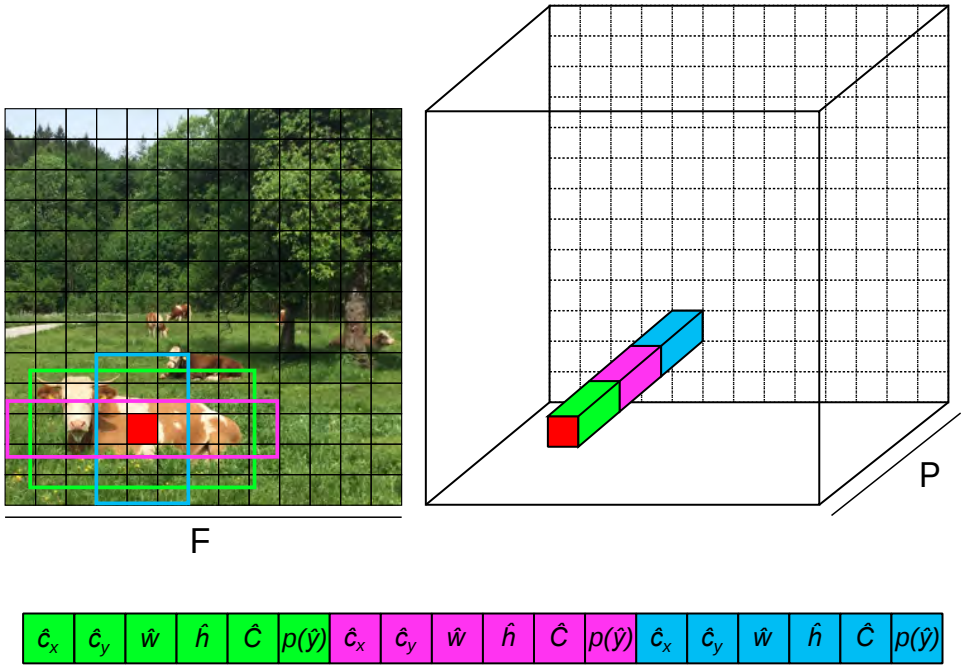


Abbildung 5.2: Der Aufbau des Vektors mit der Vorhersage für ein Pixel der Merkmalskarte. Mit nur einer Objektklasse im Trainingsdatensatz enthält der Vektor ingesamt 18 Elemente.

Das letzte Elemente zu jedem Rahmen im Vektor gibt die Wahrscheinlichkeit der Klassenzugehörigkeit $p(\hat{y})$ an, diese wird für jede der insgesamt n Klassen vorhergesagt.

Die gesamte Vorhersage für eine Merkmalskarte mit der Größe $F \times F$ hat die Dimension $F \times F \times P$, wobei P mit Hilfe der Anzahl der Objektrahmen B nach folgender Formel berechnet wird:

$$P = B \cdot (4 + 1 + n) \tag{5.1}$$

Die Berechnung der Vorhersagen des erstellten Modells erfolgt mit der Sigmoid Aktivierungsfunktion. Für das Trainieren des Netzes und die Anpassung der Gewichte wurde eine komplexe Fehlerfunktion definiert, die sowohl die Güte der Objektlokalisierung als auch die richtige Klassifikation berücksichtigt. Als Fehlerfunktion für die Klassifikation wird die Kreuzentropie (Binary Cross-Entropy, BCE) eingesetzt, welche für zwei Klassen y = 1 und y ≠ 1 folgende Werte einnimmt:

$$BCE = \begin{cases} -\log(p(\hat{y})), & \text{für y} = 1 \\ -\log(1 - p(\hat{y})), & \text{für y} \neq 1 \end{cases} \tag{5.2}$$

Die Kreuzentropie für N-Datenpunkte mit dem Zielwert y und der vorhergesagten Wahrscheinlichkeit der Klassenzugehörigkeit $p(\hat{y})$ wird wie folgt berechnet:

$$E_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(\mathrm{y}_i \cdot \log(p(\hat{\mathrm{y}}_i)) + (1 - \mathrm{y}_i) \cdot \log(1 - p(\hat{\mathrm{y}}_i))) \tag{5.3}$$

Bei einer Eingabebildgröße von z.B. 416 × 416 Pixel haben die Detektionsmerkmalskarten die Auflösungen 13 × 13 Pixel, 26 × 26 Pixel und 52 × 52 Pixel. Dies entspricht 3549 Pixel für alle drei Merkmalskarten. Da für jedes Pixel drei Objektrahmen erstellt werden, ergeben sich insgesamt 10647 Objektvorhersagen pro Bild.

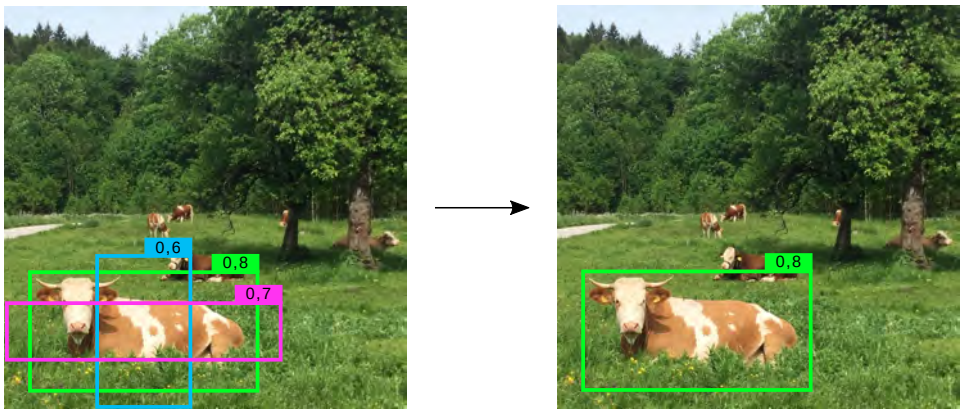


Abbildung 5.3: Das Aussortieren der überlappenden Objektrahmen nach der höchsten Objektwahrscheinlichkeit.

Um leere oder sich überlappende Objektrahmen auszusortieren, wird die Methode namens Non-Maximum Suppression angewendet (Michelucci, 2019). Dabei werden im ersten Schritt alle

Objektrahmen aussortiert, deren vorhergesagte Objektwahrscheinlichkeit $\hat{C}$ kleiner ist als 0,6. Anschließend wird für alle verbliebenen Objektrahmen der Jaccard-Koeffizient (vgl. den Abschnitt 2.4.5) berechnet. Stark überlappende Objektrahmen haben einen hohen Jaccard-Koeffizienten. Sie markieren mit hoher Wahrscheinlichkeit das gleiche Objekt. Unter den Objektrahmen mit einem Jaccard-Koeffizienten von mehr als 0,5 ist der Objektrahmen mit der höchsten Objektwahrscheinlichkeit das Ergebnis (vgl. Abbildung 5.3).

5.1.2 RetinaNet

Das neuronale Netz RetinaNet wurde in (Lin u. a., 2017b) vorgestellt. Als Gerüst des Netzes dient das CNN names ResNet-50, welches aus 50 Faltungsschichten besteht. Ähnlich wie YOLOv3 basiert dieses Netz auf dem Konzept der Objektdetektion in unterschiedlichen Generalisierungsstufen.
Bei der Entwicklung von RetinaNet wurde das Problem der ungleichen Verteilung der Klassen *Objekt* und *Hintergrund* behandelt (vgl. den Abschnitt 5.1.3). Bei RetinaNet wird ebenfalls die Sigmoid-Funktion als Aktivierungsfunktion verwendet. Als Fehlerfunktion verwendet RetinaNet jedoch eine optimierte Version der Kreuzentropie. Das Ziel dieser Optimierung ist es, beim Lernen den Einfluss der Fehler von der überrepräsentierten Klasse *Hintergrund* zu minimieren. Somit wird das Lernen mehr auf die Klasse *Objekt* fokussiert.
Für eine einfachere Notation bei den folgenden Erläuterungen wird die Formel 5.2 umgestellt. Mit der Substitution

$$p_t = \begin{cases} p(\hat{y}), & \text{für } y = 1 \\ 1 - p(\hat{y}), & \text{für } y \neq 1 \end{cases} \tag{5.4}$$

und 5.2 folgt somit:

$$BCE = -\log(p_t) \tag{5.5}$$

Die Optimierung der Kreuzentropie wird durch die zusätzlichen Hyperparameter α und γ erreicht, welche den während des Lernens berechneten Fehler anders gewichten. Beide Parameter müssen experimentell ermittelt werden.
Der Hyperparameter α kann die Werte zwischen 0 und 1 annehmen und ist für zwei Klassen wie folgt definiert:

$$\alpha_t = \begin{cases} \alpha, & \text{für } y = 1 \\ 1 - \alpha, & \text{für } y \neq 1 \end{cases} \tag{5.6}$$

Mit diesem Hyperparameter wird ein einfacher Ausgleich des Klassenungleichgewichts erreicht:

$$BCE = -\alpha_t \log(p_t) \tag{5.7}$$

Der Hyperparameter γ soll den Einfluss der überrepräsentierten Klasse *Hintergrund* auf den gesamten Fehler während des Lernens zusätzlich steuern. Die Klasse *Hintergrund* gilt als einfach zu klassifizieren. Trotz einer richtigen Klassifikation haben Vertreter dieser Klasse einen minimalen Klassifikationsfehler, der jedoch aufsummiert den gesamten Fehler verfälschen kann. Mit dem Hyperparameter γ wird erreicht, dass der Klassifikationsfehler ab einem bestimmten Schwellenwert gegen Null geht. In Abbildung 5.4 ist dargestellt, wie unterschiedliche Hyperparameter γ den Verlauf der Fehlerfunktion modifizieren.

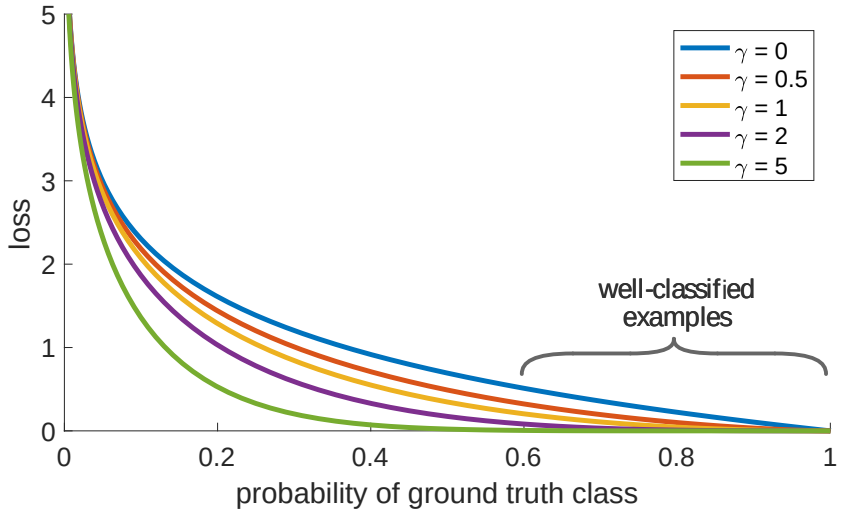


Abbildung 5.4: Die Auswirkung von unterschiedlichen γ-Werten auf den Verlauf der Fehlerfunktion (loss). Mit $\gamma = 0$ entspricht die Fehlerfunktion der Kreuzentropie wie in 5.3 (Quelle: Lin u. a. 2017b, modifiziert).

Ein optimaler Wert für den Hyperparameter α hängt vom Trainingsdatensatz und dem Hyperparameter γ ab. Je größer der Wert für den Hyperparameter γ gesetzt wurde, desto kleiner muss der Wert des Hyperparameters α gewählt werden. Anhand von Experimenten wird von den Entwicklern des Netzes der Wert 2 für den Parameter γ und der Wert 0,25 für den Hyperparameter α vorgeschlagen.
Die allgemeine Fehlerfunktion mit den Hyperparametern α und γ ergibt sich als:

$$E_{BCE} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{5.8}$$

5.1.3 Datenaufbereitung

Für das Trainieren eines neuronalen Netzes für die Objektdetektion müssen die Datensätze anders als bei einem CNN aufbereitet werden. Die Eingabebilder in den Trainings- und Testdatensätzen werden nicht mehr nur nach Klassen unterteilt.
Zu jedem Eingabebild wird eine Beschreibung der Objektrahmen erstellt, die zu diesem Bild gehören. Die Beschreibung eines Objektrahmens beinhaltet die Information über die Klasse y des Objekts, die Höhe h, die Breite w des Objektrahmens und die c_x- und c_y-Koordinaten des Rahmenzentrums im Bildkoordinatensystem. Solche Beschreibungen, die auch Label genannt

werden, werden in diesem Projekt mit der Software LabelImg erstellt.
Die Klassen bei einer Objektdetektion können grob in zwei Gruppen unterteilt werden. Zu der ersten Gruppe gehören die *Objekt*-Klassen, welche durch die oben genannten Beschreibungen definiert werden müssen. Die zweite Gruppe besteht aus der Klasse *Hintergrund*. Dieser Klasse werden automatisch alle Bildpixel zugeordnet, die sich nicht innerhalb der Objektrahmen befinden. Aus diesem Grund ist es wichtig, alle Objekte einer definierten *Objekt*-Klasse im Bild mit Objektrahmen zu markieren.
Die Einzelbaumdetektion in diesem Projekt wird nur auf eine *Objekt*-Klasse *Baum* begrenzt. Für eine feinere Klassenunterteilung liegen nicht genug Referenzdaten vor.
Da die zur Verfügung stehenden Referenzdaten mit den bekannten Baumstandorten nicht flächendeckend vorhanden sind, werden auch Bäume ohne Referenzen für das Erstellen des Training- und des Testdatensatzes verwendet (vgl. Abbildung 5.5). So werden alle Objekte der *Objekt*-Klasse *Baum* in den Eingabebildern markiert.

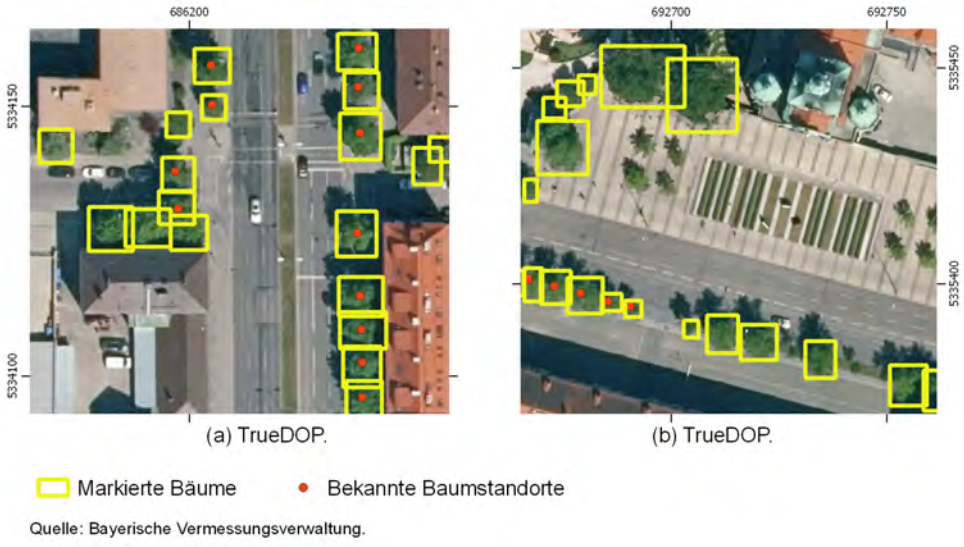


Abbildung 5.5: Bäume im Trainingsdatensatz, die mit Objektrahmen markiert sind. Nur ein Teil dieser Bäume hat Referenzen in Form von bekannten Baumstandorten.

Als Eingabedaten werden 3-Kanal-Bilder mit einer Auflösung von 416×416 Pixel verwendet.
Sowohl die Implementierung von YOLOv3 (Jocher, 2020) als auch die Implementierung von RetinaNet (Henon, 2018), die in diesem Projekt verwendet werden, benötigen als Eingabe Bilder mit einer Auflösung, die durch 32 teilbar ist. Der Grund dafür ist eine bestimmte Größe der Merkmalsfilter in den Faltungsschichten der Netze. Falls die Bilder dieser Anforderung nicht entsprechen, werden sie automatisch an die benötigte Größe angepasst. Die Anpassung erfolgt durch die Ergänzung der Eingabebilder mit Nullwerten an den Rändern der Bilder.
Aus den zur Verfügung stehenden Daten werden für die Erstellung der 3-Kanal-Bilder nur die Informationen ausgewählt, die die Baumeigenschaften besonders gut repräsentieren:

- Der Grünkanal aus den Spektralkanälen des TrueDOP, welcher die Reflektivität im grünen Lichtspektrum beinhaltet;
- das Normalisierte digitale Oberflächenmodell (nDOM);
- der Normalisierte Vegetationsindex (NDVI).

Im grünen Lichtspektrum ist die Reflektivität der Vegetation geringer als die Reflektivität der Umgebung (vgl. Abbildung 3.2). Diese Eigenschaft ermöglicht es, anhand der Reflektivitäten die Vegetation von der Umgebung zu unterscheiden. Mit dem Vegetationsindex NDVI wird der Unterschied zwischen der Vegetation und der Umgebung noch mehr verdeutlicht. Der nDOM-Kanal ist wichtig für die Detektion der Vegetation mit einer bestimmen Höhe. Der Höhenschwellenwert für die *Objekt*-Klasse *Baum* wurde auf 5 m festgelegt. Dieser Schwellenwert wird verwendet, um kleine Sträucher aus der Klasse *Baum* auszusortieren. Dafür werden beim Erstellen des Trainings- und des Testsdatensatzes die TrueDOPs mit den nDOMs der gleichen Gebiete verglichen. Somit werden nur Bäume mit einer Höhe von über 5 m mit einem Objektrahmen markiert (vgl. Abbildung 5.6). Mit diesem Schwellwert lässt sich die *Objekt*-Klasse *Baum* eindeutig definieren.

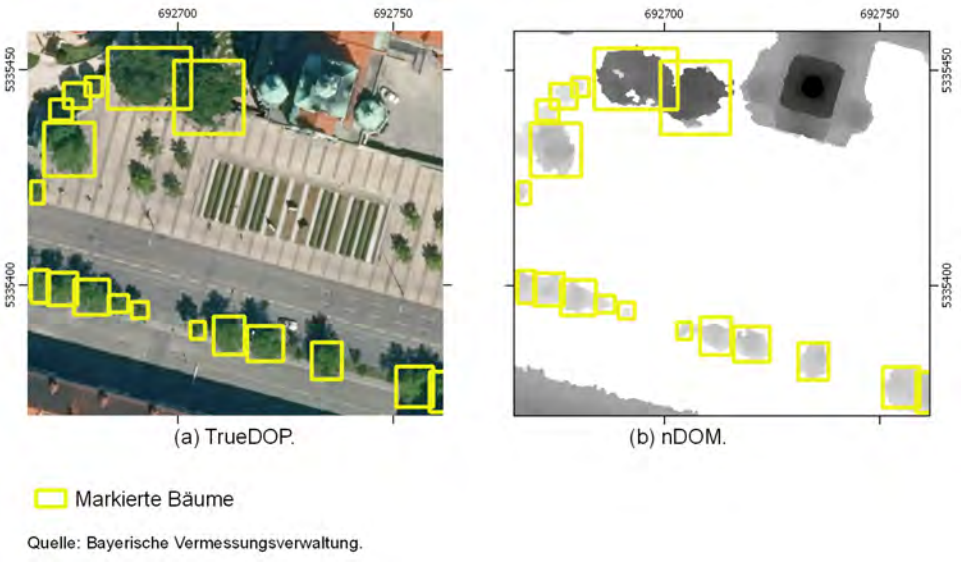


Abbildung 5.6: Bäume im Trainingsdatensatz, die mit Objektrahmen markiert sind. Das nDOM dient als Höhenreferenz bei der Erstellung des Trainings- und des Testdatensatzes. In (b) sind die Bereiche mit nDOM Werten unter 5 m weiß eingefärbt. Die Bäume in diesem Beispiel, die nicht mit einem Objektrahmen versehen sind, haben einen nDOM Wert, der kleiner als 5 m ist.

Die nDOM Werte und die NDVI Werte liegen im 32-Bit-Format mit Fließkommazahl vor. Der Wertebereich dieses Formats liegt zwischen ca. -10^{38} und 10^{38}. Die Werte können auf 6 Nachkommastellen genau angegeben werden. Da programmtechnisch bei YOLOv3 und RetinaNet nur eine Eingabe mit einer 8-Bit-Farbtiefe je Kanal möglich ist, müssen die nDOM Werte und die NDVI Werte umformatiert werden. Das 8-Bit-Format ohne Vorzeichen hat einen Wertebereich zwischen 0 und 2^8 und kann somit 256 Abstufungen darstellen. Die Nachkommastellen können bei diesem Format nicht angegeben werden.
Die nDOM Werte werden direkt aus dem 32-Bit-Format in das 8-Bit-Format umgerechnet. Durch das Umformatieren werden alle negativen Werte und die Nachkommastellen entfernt. Das Ergebnis ist eine grobe Abstufung der Höhenwerte mit einer Genauigkeit im Meterbereich.
Das direkte Umwandeln der NDVI Werte ist wegen dessen Wertebereich zwischen -1 und 1 nicht möglich. Daher werden die Werte zuerst mit dem Faktor 1000 multipliziert und danach in das 8-Bit-Format umgewandelt. Durch die Umwandlung in das 8-Bit-Format werden die negativen NDVI Werte auf 0 gesetzt. Die Werte zwischen 0 und 1000 werden auf die restlichen

255 Abstufungen verteilt. Da die Vegetation nur positive NDVI Werte aufweist, geht beim Umformatieren nur ein kleiner Teil der Information über die Vegetation verloren. Die NDVI Werte für die Vegetation werden dadurch grober abgestuft.

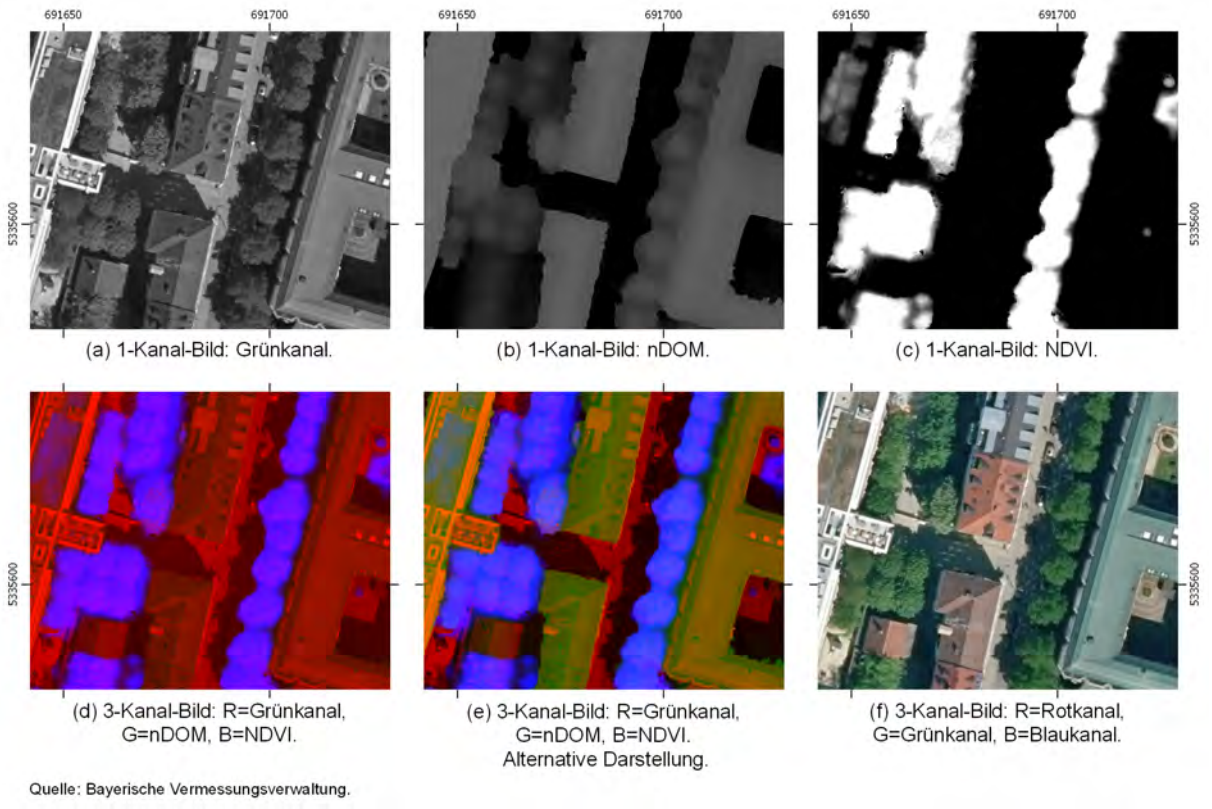


Abbildung 5.7: In (a) – (c) sind die drei einzelnen Kanäle im 8-Bit-Format abgebildet. Das 3-Kanal-Bild in Falschfarbendarstellung ist in (d) und das originale TrueDOP in (f) zu sehen. In (e) ist das 3-Kanal-Bild aus (d) in einem anderen Darstellungsformat zu sehen. So wird der nDOM-Kanal im Bild auch für das menschliche Auge sichtbar.

Ein Beispiel für das entstandene 8-Bit-Raster mit drei Kanälen ist in der Abbildung 5.7 dargestellt. Insgesamt werden 1252 Bilder für das Trainieren und das Testen der beiden Netze verwendet. Davon sind 1164 Bilder mit Objektbeschreibungen versehen. Die restlichen Bilder beinhalten nur Beispiele für die Klasse *Hintergrund*. Alle Bilder werden in drei Datensätze aufgeteilt. Der Trainings- und der Validierungsdatensatz werden während des Lernens eines Netzes verwendet. Sie beinhalten 802 bzw. 300 Bilder. Der Testdatensatz wird für eine unabhängige Bewertung des erstellten Modells nach dem Lernen eingesetzt und besteht aus 150 Bildern. Die genaue Anzahl der Objekte in jedem Datensatz ist in der Tabelle 5.1 dargestellt.

Tabelle 5.1: Datensätze für das Trainieren und das Testen der Netze YOLOv3 und RetinaNet.

Anzahl der Objekte			
Trainingdatensatz	Validierungsdatensatz	Testdatensatz	Gesamt
11698	3719	2128	17545

5.1.4 Einstellungen während der Trainingsphase

Es werden insgesamt drei Experimente mit den beiden Netzen durchgeführt.
Das Netz YOLOv3 wird zweimal trainiert. Für das erste Training werden die Gewichte des Netzes zufällig initialisiert. Beim zweiten Experiment werden auf dem Datensatz COCO vortrainierte Gewichte für die Initialisierung des Netzes eingesetzt. Während dieser beiden Experimente werden Manipulationstechniken der Datenerweiterung wie Rotation und Skalierung eingesetzt. Dabei werden die Eingabebilder vom Netz zufällig um 45° gedreht und um den Faktor 0,25 entweder vergrößert oder verkleinert.
Das Netz RetinaNet wird nur mit zufälligen Gewichten trainiert. Als Datenerweiterung wird nur die horizontale Spiegelung der Bilder verwendet.
Beide Netze werden jeweils 300 Lerniterationen, die auch Epochen genannt werden, trainiert. Dabei werden bei jedem Experiment nur die beste Ergebnisse aus allen Epochen betrachtet.
Als Optimierungstechnik für das Gradientenabstiegsverfahren wird die Adam-Optimierung eingesetzt. Die Lernrate am Anfang des Trainings wird auf 0,001 festgelegt.
Wie erfolgreich das Training der Netze ist, wird anhand von Metriken wie Präzision, Trefferquote, F_1-Maß und mittlere durchschnittliche Präzision bewertet. Bei der Objektdetektion hängen die Präzision und die Trefferquote, die für die Berechnung der gemittelten durchschnittlichen Präzision benötigt werden, vom IoU-Grenzwert ab. Für die Bewertung von YOLOv3 und RetinaNet wird ein IoU-Grenzwert von 0,5 verwendet.

5.2 Domänenanpassung

In diesem Projekt wird die Methode der ungepaarten Bildübersetzung für die Domänenanpassung erprobt. Für die Übersetzung der Bilder wird das neuronale Netz CycleGAN (Zhu u. a., 2017) eingesetzt, welches im folgenden Unterkapitel beschrieben wird.

5.2.1 CycleGAN

Im Unterschied zu den GANs, welche im Abschnitt 2.1.3 beschrieben wurden, besteht das Netz CycleGAN aus zwei Generatoren und zwei Diskriminatoren.
Die beiden Generatoren definieren zwei Abbildungsfunktionen $G : \mathcal{X} \to \mathcal{Y}$ und $F : \mathcal{Y} \to \mathcal{X}$. Die Diskriminatoren $D_\mathcal{Y}$ und $D_\mathcal{X}$ werden trainiert, um jeweils die generierten Daten $G(\mathbf{x})$ mit $\mathbf{x} \in \mathcal{X}$ und $F(\mathbf{y})$ mit $\mathbf{y} \in \mathcal{Y}$ und die realen Daten $\mathbf{x}$ und $\mathbf{y}$ richtig zu klassifizieren.
Die Erstellung der Abbildungsfunktionen der Generatoren wird durch einen zusätzlichen Regulierungsfaktor beeinflusst. Dieser Regulierungsfaktor ist eine zyklische Konsistenz (Cycle Consistency) zwischen beiden Generator-Diskriminator-Paaren. Mit dieser zyklischen Konsistenz soll erreicht werden, dass die Übersetzung eines Bildes x durch die Funktion G durch die Funktion F zurück in das Bild x übersetzt werden kann (vgl. Abbildung 5.8):

$$F(G(\mathbf{x})) \approx \mathbf{x} \qquad G(F(\mathbf{y})) \approx \mathbf{y} \tag{5.9}$$

(a) Das Eingabebild x $\in \mathcal{X}$. **(b)** Das übersetzte Bild $G(\mathrm{x})$. **(c)** Das zurück übersetzte Bild $F(G(\mathrm{x})) \approx \mathrm{x}$.

Abbildung 5.8: Die zyklische Konsistenz des Netzes CycleGAN am Beispiel der Übersetzung zwischen einem Pferdebild und einem Zebrabild (Quelle: Zhu u. a. 2017, modifiziert).

Der schematische Aufbau des Netzes ist in Abbildung 5.9 dargestellt.
Der allgemeine GAN-Lernalgorithmus wird im CycleGAN durch die Fehlerfunktion der zyklischen Konsistenz erweitert. Die Fehlerfunktion der zyklischen Konsistenz ist wie folgt definiert:

$$\mathcal{L}_{cyc}(G,F) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\|F(G(\mathbf{x})) - \mathbf{x}\|_1] + \mathbb{E}_{\mathbf{y}\sim p_{data}(\mathbf{y})}[\|G(F(\mathbf{y})) - \mathbf{y}\|_1] \tag{5.10}$$

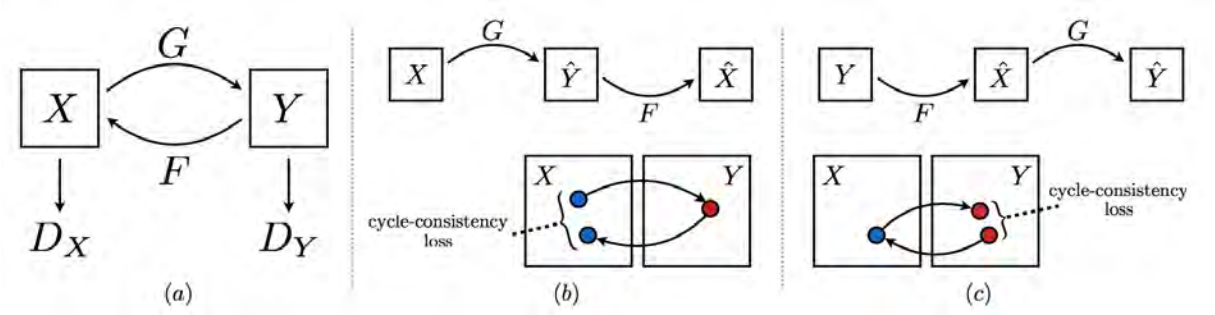


Abbildung 5.9: Der Aufbau des Netzes CycleGAN. Die Verbindung zwischen den beiden Generator-Diskriminator-Paaren in (a) sowie die schematische Erklärung der zyklischen Konsistenz des Netzes für $F(G(\mathbf{x})) \approx \mathbf{x}$ in (b) und für $G(F(\mathbf{y})) \approx \mathbf{y}$ in (c) (Quelle: Zhu u. a. 2017).

Der allgemeine Lernalgorithmus für eine ungepaarte Bildübersetzung mit dem Netz CycleGAN ist die Kombination der Fehlerfunktion der zyklischen Konsistenz mit den Wertfunktionen der beiden Generator-Diskriminator-Paaren:

$$\mathcal{L}(G,F,D_\mathcal{Y},D_\mathcal{X}) = \min_G \max_{D_\mathcal{Y}} \mathcal{L}_{GAN}(G,D_\mathcal{Y}) + \min_F \max_{D_\mathcal{X}} \mathcal{L}_{GAN}(F,D_\mathcal{X}) + \lambda\mathcal{L}_{cyc}(G,F) \tag{5.11}$$

Der Hyperparameter λ steuert den Einfluss des Fehlers der zyklischen Konsistenz auf den Gesamtfehler. Die Entwickler des Netzes haben alle Experimente mit $\lambda = 10$ durchgeführt.

5.2.2 Datenaufbereitung

Das CycleGAN wird auf 3-Kanal-Bildern trainiert. Für die Erstellung der Bilder wird die Information aus dem Nahinfrarot-, dem Rot- und dem Grünkanal der Spektralkanäle der TrueDOPs verwendet.
Diese Spektralkanäle sind ausgewählt, da sie nach der Übersetzung für die Erstellung der 3-Kanal-Bilder für die Einzelbaumdetektion zusammengesetzt werden können. Der Grünkanal wird direkt nach der Übersetzung übernommen und die NDVI Werte werden aus dem Verhältnis zwischen den Werten im Rot- und im Nahinfrarotkanal berechnet. Die nDOM Werte, welche bei der Einzelbaumdetektion verwendet werden, müssen nicht übersetzt werden.
Es werden 3-Kanal-Bilder mit einer Auflösung von 256×256 Pixel verwendet, wie in den Experimenten von Zhu u. a. (2017).
Da jedes Untersuchungsgebiete $1\,\mathrm{km}^2$ groß ist, entspricht das 361 Bildern mit einer Auflösung von 256×256 Pixel pro Gebiet. Für das Trainieren des Netzes ist diese Anzahl der Bilder nicht ausreichend. Daher werden bei der Domänenanpassung nicht nur Daten der Untersuchungsgebiete selbst, sondern auch Daten der umliegenden Gebiete verwendet. Bei der Wahl der umliegenden Gebiete wird darauf geachtet, dass die Daten von derselben Befliegungsfirma am selben Tag aufgenommen wurden.
Für die ungepaarte Bildübersetzung mit dem Netz CycleGAN werden sowohl Bilder der Ziel- als auch der Quelldomäne in Trainings- und Testdatensätze aufgeteilt. Die Größe der Datensätze für jedes Gebiet ist in der Tabelle 5.2 dargestellt.

Tabelle 5.2: Datensätze der Quell- und der Zieldomänen für die Domänenanpassung.

Gebiet	**Domäne**	**Anzahl der Bilder**	
		Trainingdatensatz	Testdatensatz
Augsburg	Zieldomäne	3300	300
München	Quelldomäne	3670	330
Nürnberg	Zieldomäne	3300	300
Bad Kissingen	Zieldomäne	3300	300
Windischeschenbach	Zieldomäne	3300	300
Stockdorf	Quelldomäne	3670	330

5.2.3 Einstellungen während der Trainingsphase

Insgesamt werden vier CycleGAN Modelle für die Domänenanpassung erstellt. Die Gewichte des Netzes werden zufällig initialisiert. Jedes Modell wird 300 Epochen trainiert. Innerhalb der ersten 150 Epochen wird die Lernrate gleich gehalten und während der nächsten 150 Epochen wird die Lernrate linear bis auf null verringert. Für das Gradientenabstiegsverfahren wird die Adam-Optimierung verwendet. Die Lernrate am Anfang des Trainings wird auf 0,0002 festgelegt. Diese Einstellungen entsprechen den Experimenten, welche die Entwickler des Netzes durchgeführt haben. Sie sind ebenfalls in der Version des Netzes CycleGAN (Zhu u. a., 2018), die in diesem Projekt verwendet wird, implementiert.
Da keine verlässlichen Validierungsmetriken für die Bewertung der Ergebnisse von CycleGAN existieren, wird für dieses Projekt eine auf den Anwendungsfall bezogene Auswertungsmethode vorgeschlagen. Der Fokus bei der Übersetzung der Bilder in diesem Projekt liegt auf der Vegetation.

Deswegen werden in jedem Untersuchungsgebiet Bereiche mit Laubbäumen für die Auswertung ausgewählt. Die Verteilungen der spektralen Information dieser Bereiche der Zieldomänen werden vor und nach der Übersetzung mit der Verteilung der Quelldomänen verglichen.
Für den Vergleich der Verteilungen wird der Fréchet-Abstand verwendet. Der Fréchet-Abstand d für zwei univariate Verteilungen $H(\mathbf{x})$ und $P(\mathbf{y})$ ist wie folgt definiert (Dowson und Landau, 1982):

$$d^2(H(\mathbf{x}), P(\mathbf{y})) = (\mu_\mathbf{x} - \mu_\mathbf{y})^2 + (\sigma_\mathbf{x} - \sigma_\mathbf{y})^2 \tag{5.12}$$

Dabei sind $\mu_\mathbf{x}$ und $\mu_\mathbf{y}$ die Mittelwerte und $\sigma_\mathbf{x}$ und $\sigma_\mathbf{y}$ die Standardabweichungen von $H(\mathbf{x})$ und $P(\mathbf{y})$ entsprechend.

5.3 Validierung mit K-Nearest-Neighbors

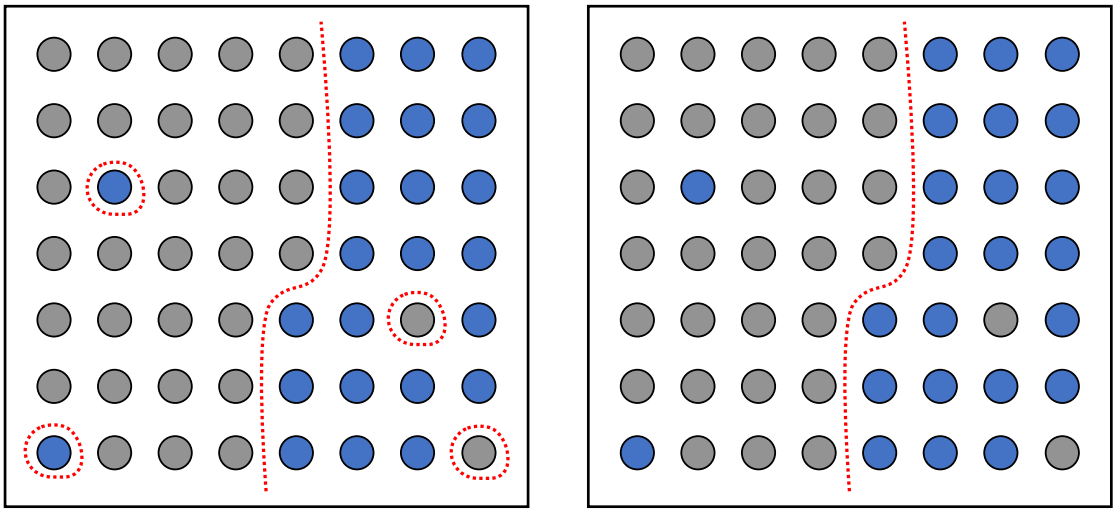
Das Detektionsergebnis wird nicht nur anhand des Validierungs- und Testdatensatzes bewertet. Das erstellte Detektionsmodell mit dem besten Ergebnis wird auch auf sechs verschiedene Gebiete ohne Labels angewendet. Die Größe jedes Gebiets ist $1\,\mathrm{km}^2$. Es wird dabei untersucht, wie groß der Anteil der detektierten Vegetation an der Gesamtvegetation mit einer Höhe von über 5 m ist. Außerdem wird die Fehlerquote für die einzelnen Gebiete bestimmt.
Bei dieser Untersuchung wird die Zusammensetzung der detektierten Bounding Boxen pixelweise ausgewertet, um mögliche Fehldetektionen zu erkennen. Für diese Auswertung wird die Information über die Gesamtvegetation benötigt. Die gesamte Vegetationsfläche in den einzelnen Gebieten kann nur bedingt mit bestehenden Datensätzen wie dem amtlichen topographisch-kartographischen Informationssystem (ATKIS) ermittelt werden (LDBV, 2022). Solche Datensätze geben nur einen allgemeinen Überblick über die Landbedeckung, ohne feine Strukturen zu berücksichtigen.
Aus diesem Grund wird für eine flächendeckende Ermittlung der Vegetationsfläche ein weiteres Klassifikationsverfahren namens K-Nearest-Neighbors (KNN) eingesetzt.

5.3.1 K-Nearest-Neighbors (KNN)

KNN ist ein nicht-parametrisches Klassifizierungsverfahren und zählt zu den Methoden des überwachten Lernens. Bei dem KNN-Verfahren wird die Zugehörigkeit eines Datenpunktes zu einer Klasse über die naheliegenden Punkte, k nächsten Nachbarn, bestimmt. Für k gilt $1 < k < m$, wobei m die Anzahl der Einträge im Trainingsdatensatz ist (Diawara, 2019). Je häufiger eine Klasse unter den k nächsten Nachbarn eines Datenpunktes vorkommt, desto wahrscheinlicher ist es, dass dieser Datenpunkt zu dieser Klasse gehört. Haben mehrere Klassen dieselbe Häufigkeit, wird eine dieser Klassen zufällig ausgewählt. Als Abstandsfunktion wird meistens die euklidische Distanz verwendet. Die Distanz d zwischen zwei Datenpunkten p und q in einem n-dimensionalen Merkmalsraum wird wie folgt berechnet:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{5.13}$$

Beim KNN Verfahren allgemein ist zu beachten, dass das Ergebnis des Trainings von der Einstellung des k-Wertes abhängig ist (Gareth u. a., 2013). Bei einem zu kleinen k-Wert wird das Modell stark an den Trainingsdatensatz angepasst, man redet dabei von einer Überanpassung des Modells. Im Gegensatz dazu bildet das Modell bei einem zu großen k-Wert die Zusammenhänge in den Daten nur grob ab. Dieses Phänomen wird als Unteranpassung bezeichnet.
Für das Eliminieren sowohl einer Überanpassung als auch einer Unteranpassung werden die Genauigkeitswerte auf den Trainings- und Validierungsdatensätzen verglichen. Eine viel größere Genauigkeit auf dem Trainingsdatensatz als auf dem Validierungsdatensatz bedeutet eine Überanpassung, und umgekehrt eine Unteranpassung.
Das Beispiel in Abbildung 5.10 zeigt zwei KNN-Klassifikationsmodelle, welche mit unterschiedlichen k-Werten trainiert wurden. Die Entscheidungsgrenze (rot gestrichelte Linie) repräsentiert die Trennung zwischen zwei Klassen (blaue und graue Punkte). Mit dem Wert $k = 1$ in **(a)** neigt das Klassifikationsmodell zur Überanpassung, da einzelne vermutliche Ausreißer der beiden Klassen bei der Definition der Entscheidungsgrenze berücksichtigt wurden. Das Klassifikationsmodell in **(b)**, welches mit dem Wert $k = 4$ trainiert wurde, hat eine optimalere Entscheidungsgrenze definiert, die robust gegenüber Ausreißern in den Daten ist.



(a) Das Klassifikationsmodell bei $k = 1$. **(b)** Das Klassifikationsmodell bei $k = 4$.

Abbildung 5.10: Zwei KNN-Klassifikationsmodelle, welche mit unterschiedlichen k-Werten trainiert wurden. Die Entscheidungsgrenze (rot gestrichelte Linie) repräsentiert die Trennung zwischen zwei Klassen (blaue und graue Punkte).

5.3.2 Datenaufbereitung

Für die KNN-Klassifikation werden eigene Trainings- und Validierungsdatensätze für alle Untersuchungsgebiete erstellt. Es wird pro Gebiet ein KNN-Modell trainiert und validiert.
Bei der Klassifikation wird zwischen der Klasse *Baum* und der Klasse *Kein Baum* unterschieden. Einheitlich zu der Objektdetektion wird für die Klasse *Baum* nur die Vegetation mit einer relativen Höhe von über 5 m ausgewählt. Abhängig von der Beschaffenheit jedes Gebietes wird diese Klasse sowohl Laub- als auch Nadelbäume enthalten. Der Klasse *Kein Baum* werden sowohl Bauten wie Gebäudedächer und Straßen, als auch Ackerflächen verschiedener Art und

Wiesen zugeordnet. Als Referenz bei der Erstellung der Datensätze werden die Digitalen True Orthophotos und ergänzend auch das ATKIS Basis-DLM dienen.
Die Trainings- und Validierungsdatensätze werden für jedes Untersuchungsgebiet polygonweise in QGIS ausgewählt und in Form von Vektordaten gespeichert. Im Anschluss werden die Vektordaten mit den Rasterdaten der Untersuchungsgebiete verschnitten, um die spektrale Information beider Klassen für jedes Gebiet zu extrahieren. Die für das Training und die Validierung verwendeten Datensätze sind in der Tabelle 5.3 aufgelistet.
Jedes KNN-Model wird auf den Rasterdaten mit jeweils 6 Kanälen trainiert. Außer den in den TrueDOPs zur Verfügung stehenden Spektralkanälen Rot, Grün, Blau und Nahinfrarot werden als fünfter Kanal nDOM Werte und als sechster Kanal NDVI Werte hinzugefügt.
Die Rasterdaten werden im 32-Bit-Format mit Fließkommazahl gespeichert, was die Verwendung von Nachkommastellen und negativen Werten ermöglicht. Damit können die nDOM Werte und die NDVI Werte mit einer höhere Genauigkeit angegeben werden.

Tabelle 5.3: Datensätze für die K-Nearest-Neighbors-Klassifikation mit der Anzahl der Pixel in Millionen.

Validierungsgebiet	**Anzahl der Pixel [Millionen]**					
	Training			Validierung		
	Baum	*Kein Baum*	$\sum$	*Baum*	*Kein Baum*	$\sum$
Augsburg	1,03	1,22	2,25	0,39	0,33	0,72
München	1,01	1,17	2,18	0,31	0,37	0,68
Nürnberg	1,14	1,17	2,31	0,38	0,34	0,74
Bad Kissingen	1,18	1,07	2,25	0,32	0,36	0,68
Windischeschenbach	1,02	1,17	2,19	0,36	0,34	0,70
Stockdorf	1,08	1,13	2,21	0,32	0,34	0,66

5.3.3 Einstellungen während der Trainingsphase

Die K-Nearest-Neighbors-Klassifikation wird mit den Standardeinstellungen der Implementierung aus der Scikit-learn-Bibliothek (vgl. Tabelle 5.4) durchgeführt. Die Anzahl der nächsten Nachbarn k ist auf 5 festgelegt und als Abstandsfunktion wird die euklidische Distanz verwendet. Die Auswertung erfolgt pixelweise.

Tabelle 5.4: Wichtigste Standardeinstellungen der K-Nearest-Neighbors Implementierung aus der Scikit-learn-Bibliothek.

Parameter	**Bedeutung**
`n_neighbors=5`	Anzahl der benachbarten Punkte
`weights='uniform'`	Alle Punkte aus der Nachbarschaft sind gleich gewichtet
`p=2`	Einstellung für die verwendete Metrik
`metric='minkowski'`	Mit `p=2` entspricht die Minkowski Distanz der euklidischen Distanz (vgl. 5.13)

Insgesamt werden sechs Klassifikationsmodelle erstellt. Jedes Modell wird mit Metriken wie der Gesamtgenauigkeit und dem Kappa-Koeffizient bewertet. Die Validierungsmetriken werden für

den Trainingsdatensatz und für den Testdatensatz berechnet, um sowohl eine Überanpassung als auch eine Unteranpassung auszuschließen. Dabei wird die Vorhersage des Modells $\hat{y}$ mit den vordefinierten y_{train} und y_{test} verglichen.
Im nächsten Schritt werden die Klassifikationsmodelle auf die Untersuchungsgebiete angewendet. Somit hat das Klassifikationsergebnis für jedes Untersuchungsgebiet auch die Größe von $1\,km^2$ oder 5000×5000 Pixel.
Anschliessend wird jedes Klassifikationsergebnis mit dem nDOM des gleichen Gebietes pixelweise verglichen. Bei diesem Vergleich werden die Pixelwerte des Klassifikationsergebnises von der Klasse *Baum* auf die Klasse *Kein Baum* umgeschrieben, falls die nDOM Werte dieser Pixel kleiner als 5 m sind. Damit wird versucht, die Präzision der Klassifikationsergebnisse für die Klasse *Baum* zusätzlich zu verbessern.
Um diese Korrektur auszuwerten, wird die korrigierte Vorhersage des Modells $\hat{y}_{nDOM}$ mit dem vordefinierten y_{test} verglichen. Das Ergebnis des Vergleichs wird anhand der Konfusionsmatrizen betrachtet.
Die Gesamtvegetation, die für die Validierung der Detektionsergebnisse benötigt wird, wird anhand der KNN-Klassifikation nach der Korrektur ermittelt.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Experimente vorgestellt, welche im Kapitel 5 beschrieben wurden.

6.1 Einzelbaumdetektion

Wie in Abschnitt 5.1.4 beschrieben wurde, werden insgesamt drei Experimente mit den Netzen YOLOv3 und RetinaNet durchgeführt. Eine zusammenfassende Beschreibung der Experimente ist in Tabelle 6.1 dargestellt.

Tabelle 6.1: Die Zusammenfassung der durchgeführten Experimente mit den neuronalen Netzen YOLOv3 und RetinaNet.

Modell	**Netz**	**Gewichte**		**Datenerweiterung**		
		Zufällig	COCO	Rotation	Skalierung	Spiegelung
yolov3_{random}	YOLOv3	✓	✗	✓	✓	✗
yolov3_{coco}	YOLOv3	✗	✓	✓	✓	✗
$\text{retinanet}_{random}$	RetinaNet	✓	✗	✗	✗	✓

Das laut Validierungsmetriken beste Ergebnis wurde vom Detektionsmodell yolov3_{coco} erreicht (vgl. Tabelle 6.2). Dieses Modell wird im weiteren Verlauf für die unabhängige Auswertung der Detektionsergebnisse eingesetzt. Dabei wird das Modell auf sechs ausgewählte Untersuchungsgebiete angewendet, um das Detektionsergebnis unabhängig vom Testdatensatz zu bewerten.
Ein Beispiel für das Detektionsergebnis auf dem Testdatensatz ist in Abbildung 6.1 zu sehen.

Tabelle 6.2: Ergebnisse der Auswertung der erstellten Detektionsmodelle angewendet auf den Validierungsdatensatz (*ValD*) und auf den Testdatensatz (*TestD*).

Modell	**Datensatz**	**Präzision**	**Trefferquote**	**F_1**	**mAP**
yolov3_{random}	*ValD*	0,527	0,863	0,654	0,809
	TestD	0,528	0,879	0,660	0,822
yolov3_{coco}	*ValD*	0,704	0,864	0,776	0,847
	TestD	0,725	0,876	0,793	0,869
$\text{retinanet}_{random}$	*ValD*	0,152	0,957	0,262	0,852
	TestD	0,161	0,959	0,276	0,866

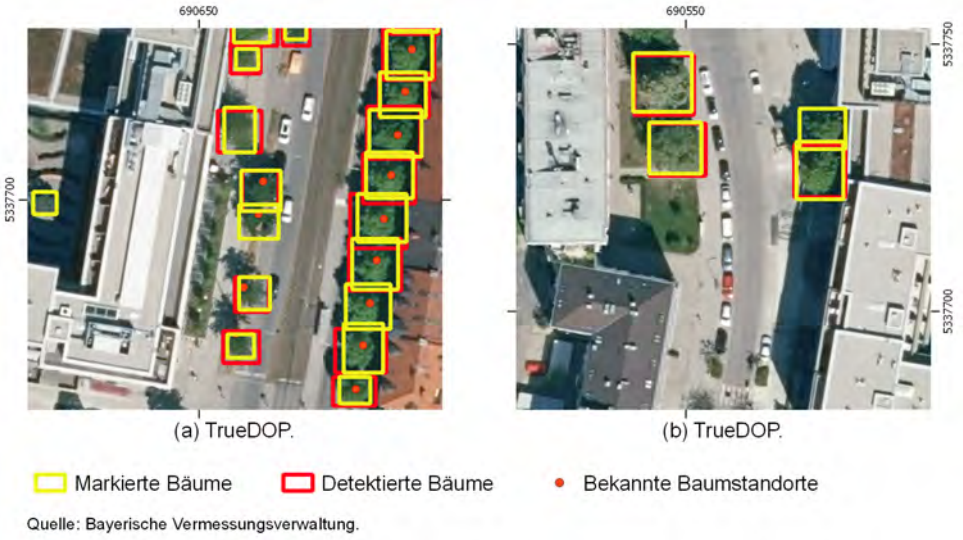


Abbildung 6.1: Ein Vergleich zwischen detektierten Bäumen und im Testdatensatz markierten Bäumen sowie deren Referenzen in Form von bekannten Baumstandorten.

6.2 Domänenanpassung

Insgesamt werden vier Domänenanpassungen durchgeführt. Welche Kombinationen der Ziel- und Quelldomänen bei der Bildübersetzung verwendet werden, ist in Tabelle 6.3 dargestellt.

Tabelle 6.3: Vier CycleGAN Modelle für die Domänenanpassung.

Modell	**Quelldomäme**	**Zieldomäne**
cycle_{m2a}	München	Augsburg
cycle_{m2n}	München	Nürnberg
cycle_{s2b}	Stockdorf	Bad Kissingen
cycle_{s2w}	Stockdorf	Windischeschenbach

Wie erfolgreich die Domänenanpassung war, wird sowohl durch die visuelle Betrachtung der Ergebnisse als auch anhand des Fréchet-Abstands bestimmt. Der Fréchet-Abstand zwischen Quell- und Zieldomänen vor und nach der Domänenanpassung ist in Tabelle 6.4 dargestellt. Als Anhaltspunkt bei dem Vergleich der berechneten Werten soll der Fréchet-Abstand zwischen München und Stockdorf dienen. Beide Gebiete wurden am gleichen Tag von derselben Befliegungsfirma aufgenommen. In den Abbildungen 6.2 bis 6.4 sind Beispiele für die Domänenanpassung in den Untersuchungsgebieten Augsburg, Nürnberg und Bad Kissingen zu sehen.

Tabelle 6.4: Der Fréchet-Abstand zwischen den Reflektivitäten im Nahinfrarot- (NIR), im Rot- (R) und im Grünkanal (G) der Quell- und Zieldomänen vor und nach der Domänenanpassung (DA). Als Anhaltspunkt bei dem Vergleich der berechneten Werte soll der Fréchet-Abstand zwischen München und Stockdorf dienen.

Quelldomäme	**Zieldomäne**	**Fréchet-Abstand**					
		Vor DA			Nach DA		
		NIR	*R*	*G*	*NIR*	*R*	*G*
München	Augsburg	19,37	1,36	7,16	12,56	4,78	6,25
München	Nürnberg	13,04	25,32	22,68	5,72	4,89	2,15
Stockdorf	Bad Kissingen	26,62	3,38	3,29	17,78	11,63	12,17
Stockdorf	Windischeschenbach	26,26	25,83	26,45	11,66	6,10	8,94
München	Stockdorf	9,91	8,12	7,51	–	–	–

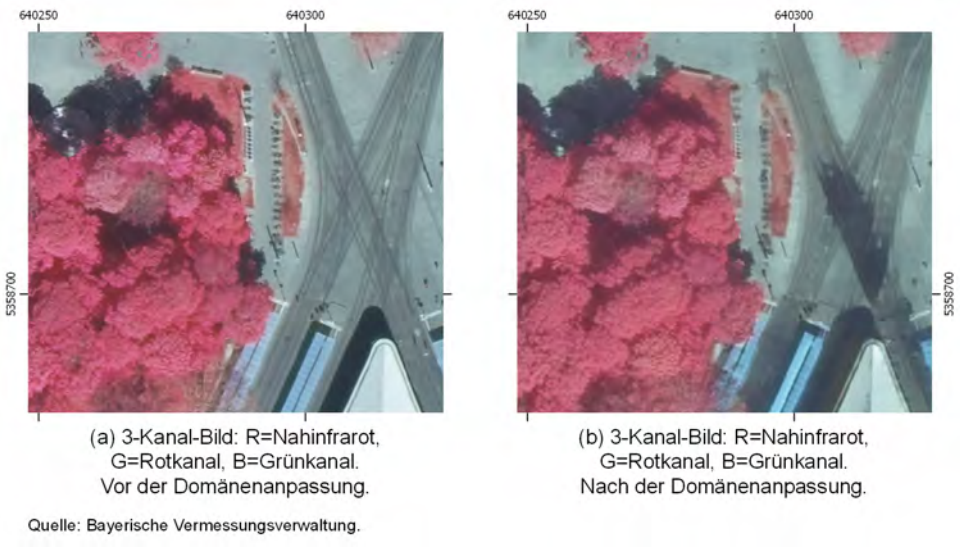


Abbildung 6.2: Das Ergebnis der Domänenanpassung im Untersuchungsgebiet Augsburg mit dem Modell cycle_{m2a}.

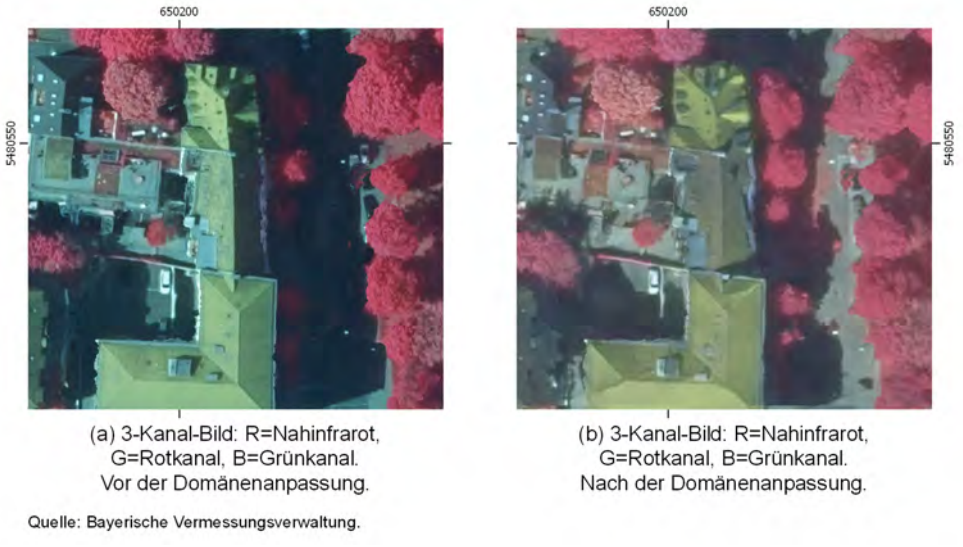


Abbildung 6.3: Das Ergebnis der Domänenanpassung im Untersuchungsgebiet Nürnberg mit dem Modell cycle_{m2n}.

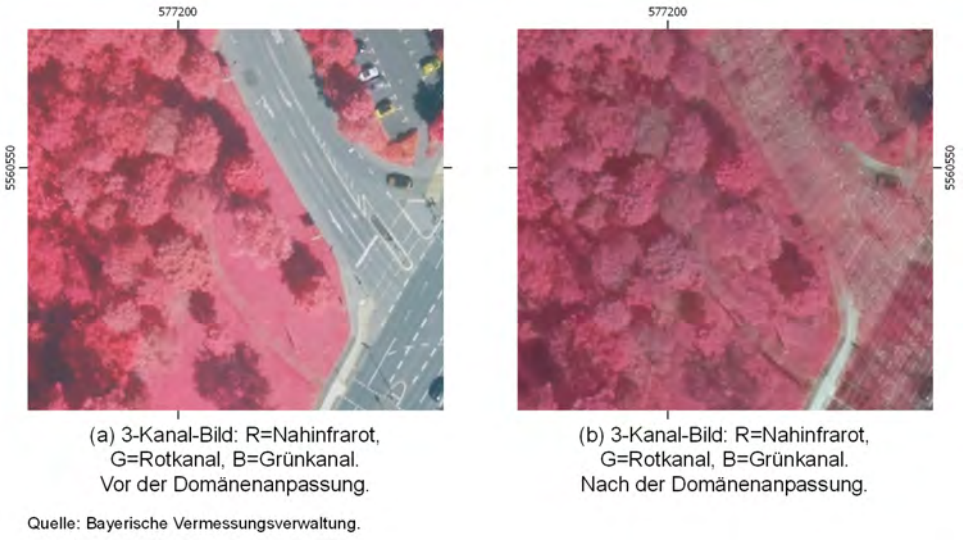


Abbildung 6.4: Das Ergebnis der Domänenanpassung im Untersuchungsgebiet Bad Kissingen mit dem Modell cycle_{s2b}.

6.3 Validierung mit K-Nearest-Neighbors

In diesem Abschnitt werden zuerst die Ergebnisse der K-Nearest-Neighbors-Klassifikation vorgestellt. Im nächsten Schritt wird anhand von dieser Klassifikation das Detektionsergebnis validiert. Das Ergebnis der Validierung wird sowohl vor als auch nach der Domänenanpassung präsentiert.

6.3.1 K-Nearest-Neighbors-Klassifikation

Das Ergebnis der KNN-Klassifikation für die sechs Untersuchungsgebiete ist in Tabelle 6.5 dargestellt.

Tabelle 6.5: Die Auswertung der Klassifikationsergebnisse für die sechs Untersuchungsgebiete.

Modell	**Untersuchungsgebiet**	**TA**$_{train}$	**TA**$_{test}$	**Kappa-Koeffizient**
knn_{Aug}	Augsburg	$> 0,999$	$0,999$	$0,997$
knn_{Mue}	München	$0,999$	$0,998$	$0,995$
knn_{Nue}	Nürnberg	$> 0,999$	$0,997$	$0,994$
knn_{BaK}	Bad Kissingen	$> 0,999$	$> 0,999$	$> 0,999$
knn_{Win}	Windischeschenbach	$> 0,999$	$0,994$	$0,988$
knn_{Sto}	Stockdorf	$> 0,999$	$> 0,999$	$> 0,999$

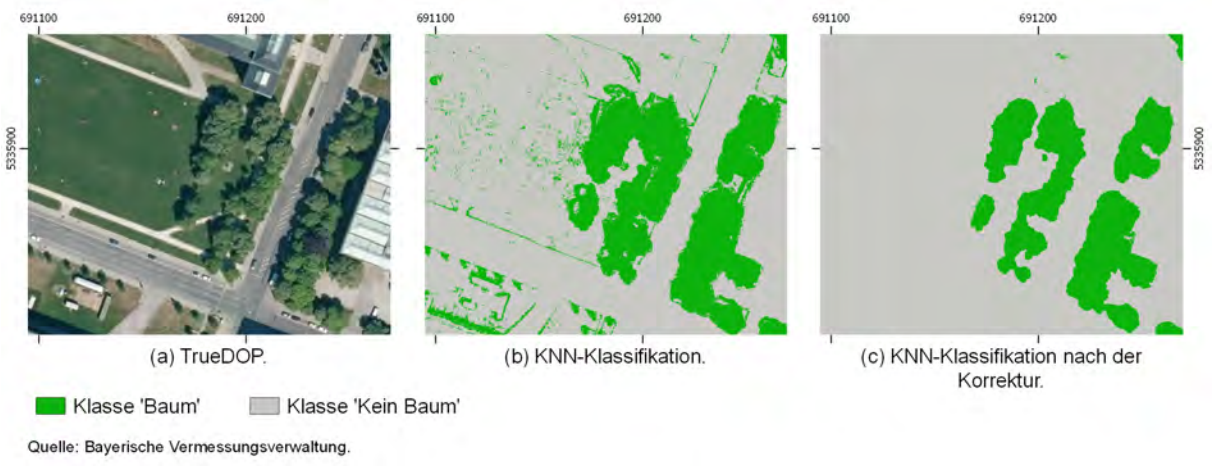


Abbildung 6.5: Das KNN-Klassifikationsergebnis vor der Korrektur (b) und nach der Korrektur (c) sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet München.

Um das Ergebnis der Korrektur der Klassifikationsergebnisse durch den Vergleich mit den nDOM Werten besser bewerten zu können, werden die Konfusionsmatrizen vor und nach der Korrektur erstellt. Die Gegenüberstellung der beiden Konfusionsmatrizen für jedes Untersuchungsgebiet ist in den Tabellen 6.7 bis 6.12 zu finden.
Die Zahl in der zweiten Zeil der ersten Spalte repräsentiert den Teil der Klasse *Kein Baum*, welcher als die Klasse *Baum* klassifiziert wurde (*false positive*, FP). Die Zahl in der ersten Zeile der zweiten Spalte repräsentiert den Teil der Klasse *Baum*, welcher als die Klasse *Kein Baum* klassifiziert wurde (*false negativ*, FN).

Tabelle 6.6: Die anhand der korrigierten KNN-Klassifikation ermittelte Gesamtvegetation (GV) mit einer Höhe von über 5 m für die sechs Untersuchungsgebiete.

Modell	**Untersuchungsgebiet**	**GV in %**	**GV in Pixel**
knn_{Aug}	Augsburg	9,7	2423234
knn_{Mue}	München	10,1	2513590
knn_{Nue}	Nürnberg	16,5	4115736
knn_{BaK}	Bad Kissingen	47,0	11632099
knn_{Win}	Windischeschenbach	27,3	6831074
knn_{Sto}	Stockdorf	39,3	9817261

In Abbildung 6.5 ist das KNN-Ergebnis vor und nach der Korrektur mit dem TrueDOP des gleichen Bereichs im Untersuchungsgebiet gegenübergestellt.
Die Gesamtvegetation für jedes Untersuchungsgebiet wurde anhand der KNN-Klassifikation nach der Korrektur ermittelt. Die Ergebnisse dieser Berechnung ist in Tabelle 6.6 dargestellt.

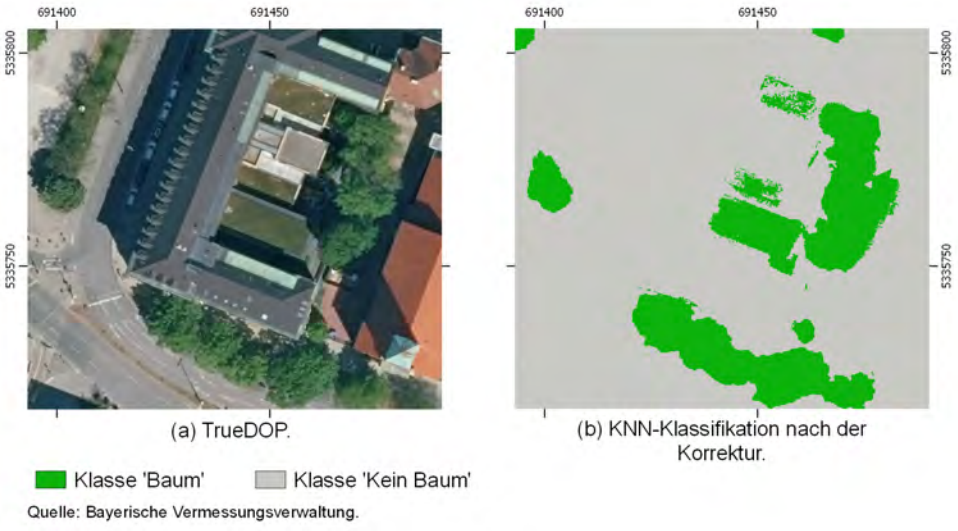


Abbildung 6.6: Das KNN-Klassifikationsergebnis nach der Korrektur (b) sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet München. Außer Bäume wurden auch begrünte Dachflächen, welche höher als 5 m sind, als Klasse *Baum* klassifiziert.

Tabelle 6.7: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet Augsburg.

(a) Vor der Korrektur.

		Vorhergesagte $\hat{y}$	
		Baum	**Kein Baum**
Vordefinierte y_{test}	**Baum**	TP 387533	FN 399
	Kein Baum	FP 669	TN 329873

(b) Nach der Korrektur.

		Vorhergesagte $\hat{y}_{nDOM}$	
		Baum	**Kein Baum**
Vordefinierte y_{test}	**Baum**	TP 387512	FN 420
	Kein Baum	FP 32	TN 330510

Tabelle 6.8: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet München.

(a) Vor der Korrektur.

		Vorhergesagte $\hat{y}$	
		Baum	**Kein Baum**
Vordefinierte y_{test}	**Baum**	TP 304180	FN 443
	Kein Baum	FP 1105	TN 365554

(b) Nach der Korrektur.

		Vorhergesagte $\hat{y}_{nDOM}$	
		Baum	**Kein Baum**
Vordefinierte y_{test}	**Baum**	TP 303932	FN 691
	Kein Baum	FP 1	TN 366658

Tabelle 6.9: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet Nürnberg.

(a) Vor der Korrektur.

Vordefinierte y_{test} \ Vorhergesagte $\hat{y}$	**Baum**	**Kein Baum**
Baum	TP 376804	FN 95
Kein Baum	FP 1934	TN 338227

(b) Nach der Korrektur.

Vordefinierte y_{test} \ Vorhergesagte $\hat{y}_{nDOM}$	**Baum**	**Kein Baum**
Baum	TP 376704	FN 195
Kein Baum	FP 100	TN 340061

Tabelle 6.10: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet Bad Kissingen. Wie zu sehen ist, hat sich das Klassifikationsergebnis nach der Korrektur nicht verändert.

(a) Vor der Korrektur.

Vordefinierte y_{test} \ Vorhergesagte $\hat{y}$	**Baum**	**Kein Baum**
Baum	TP 321362	FN 8
Kein Baum	FP 48	TN 363214

(b) Nach der Korrektur.

Vordefinierte y_{test} \ Vorhergesagte $\hat{y}_{nDOM}$	**Baum**	**Kein Baum**
Baum	TP 321362	FN 8
Kein Baum	FP 48	TN 363214

Tabelle 6.11: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet Windischeschenbach.

(a) Vor der Korrektur.

Vordefinierte $\mathbf{y}_{test}$ \ Vorhergesagte $\hat{\mathbf{y}}$	**Baum**	**Kein Baum**
Baum	TP 360458	FN 7
Kein Baum	FP 4123	TN 337479

(b) Nach der Korrektur.

Vordefinierte $\mathbf{y}_{test}$ \ Vorhergesagte $\hat{\mathbf{y}}_{nDOM}$	**Baum**	**Kein Baum**
Baum	TP 360448	FN 17
Kein Baum	FP 0	TN 341602

Tabelle 6.12: Die Konfusionsmatrizen des Klassifikationsergebnisses vor der Korrektur **(a)** und nach der Korrektur **(b)** im Untersuchungsgebiet Stockdorf.

(a) Vor der Korrektur.

Vordefinierte $\mathbf{y}_{test}$ \ Vorhergesagte $\hat{\mathbf{y}}$	**Baum**	**Kein Baum**
Baum	TP 320433	FN 62
Kein Baum	FP 25	TN 343300

(b) Nach der Korrektur.

Vordefinierte $\mathbf{y}_{test}$ \ Vorhergesagte $\hat{\mathbf{y}}_{nDOM}$	**Baum**	**Kein Baum**
Baum	TP 320428	FN 67
Kein Baum	FP 12	TN 343313

6.3.2 Validierung vor der Domänenanpassung

Das Detektionsergebnis des Modells yolov3$_{coco}$ angewendet auf die Bilder mit der Auflösung 416×416 Pixel vor der Domänenanpassung ist in Tabelle 6.13 dargestellt.
Durch den Vergleich des Detektionsergebnisses mit der KNN-Klassifikation wurde die Anzahl der detektierten Vegetationspixel und der Anteil der detektierten Vegetation an der Gesamtvegetation ermittelt. Außerdem wurde die Zusammensetzung der detektierten Bounding Boxen pixelweise ausgewertet. Die Bounding Boxen ohne Vegetationspixel werden als Fehldetektionen bezeichnet. In den Abbildungen 6.7 bis 6.11 sind Beispiele für die Objektdetektion im Vergleich mit der KNN-Klassifikation zu sehen.

Tabelle 6.13: Das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung. Dabei wurde der Anteil der detektierten Vegetation (DV) an der Gesamtvegetation mit einer Höhe von über 5 m sowie die Anzahl der Bounding Boxen mit Vegetation BB$_{richtig}$ und ohne Vegetation BB$_{falsch}$ ermittelt. Die Bounding Boxen BB$_{falsch}$ enthalten laut KNN-Klassifikation keine Vegetationspixel und werden daher als Fehldetektion betrachtet.

Untersuchungsgebiet	**BB$_{richtig}$**	**BB$_{falsch}$**	**DV in %**	**DV in Pixel**
Augsburg	921	55	49, 1	1189962
München	800	40	47, 5	1193438
Nürnberg	1325	28	33, 5	1377547
Bad Kissingen	1832	31	22, 3	2598965
Windischeschenbach	2292	32	26, 0	1775821
Stockdorf	2316	38	22, 0	2134928

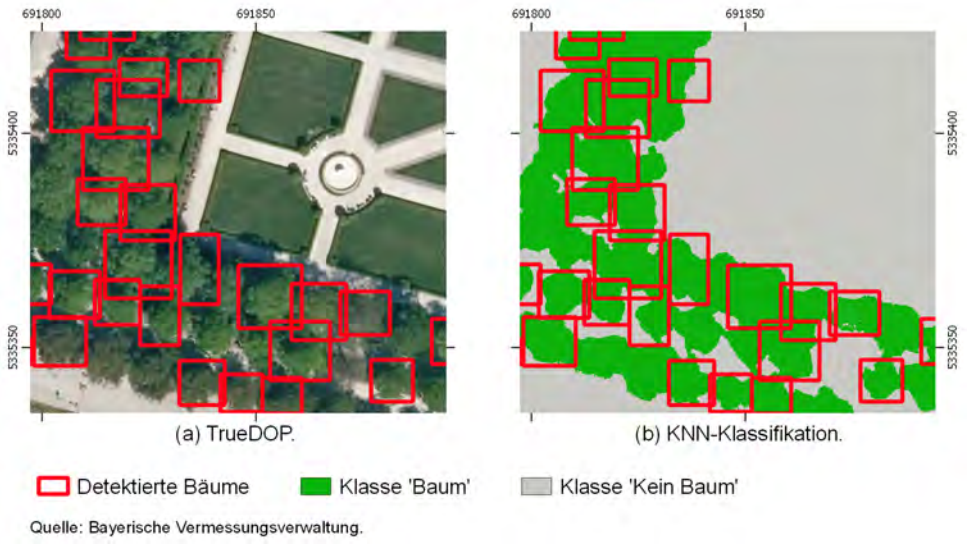


Abbildung 6.7: Ein Beispiel für das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet München.

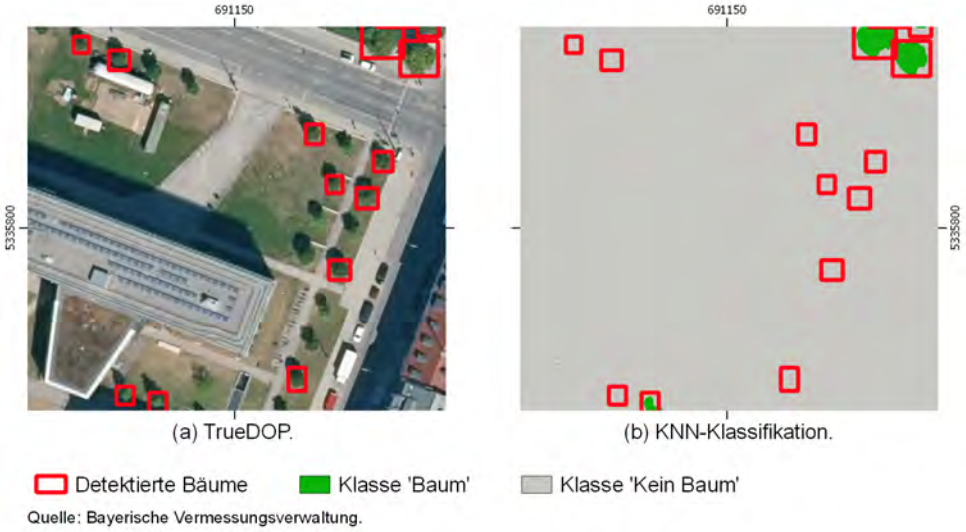


Abbildung 6.8: Ein weiteres Beispiel für das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet München.

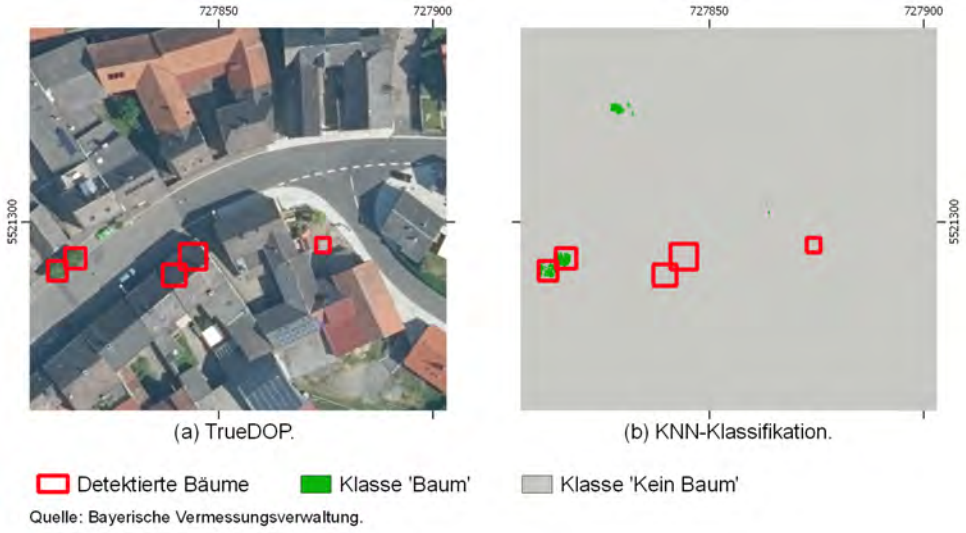


Abbildung 6.9: Das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Windischeschenbach.

Abbildung 6.10: Das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Augsburg.

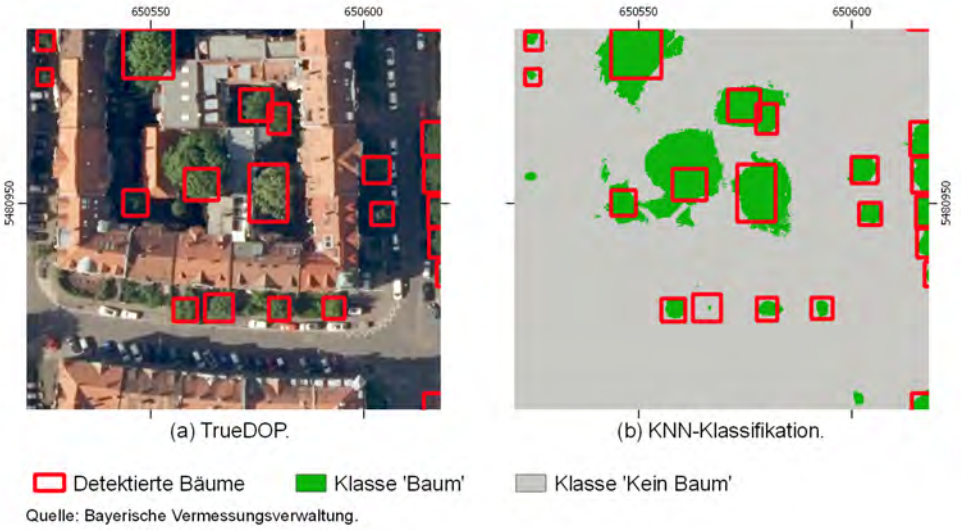


Abbildung 6.11: Das Detektionsergebnis des Modells yolov3$_{coco}$ vor der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Nürnberg.

6.3.3 Validierung nach der Domänenanpassung

Das Detektionsergebnis des Modells yolov3$_{coco}$ angewendet auf die Bilder mit der Auflösung 416×416 Pixel nach der Domänenanpassung ist in Tabelle 6.14 dargestellt.
In den Abbildungen 6.13 bis 6.14 sind Beispiele für die Objektdetektion im Vergleich mit der KNN-Klassifikation zu sehen.

Tabelle 6.14: Das Detektionsergebnis des Modells yolov3$_{coco}$ nach der Domänenanpassung. Dabei wurde der Anteil der detektierten Vegetation (DV) an der Gesamtvegetation mit einer Höhe von über 5 m sowie die Anzahl der Bounding Boxen mit Vegetation BB$_{richtig}$ und ohne Vegetation BB$_{falsch}$ ermittelt. Die Bounding Boxen BB$_{falsch}$ enthalten laut KNN-Klassifikation keine Vegetationspixel und werden daher als Fehldetektion betrachtet.

Untersuchungsgebiet	**BB$_{richtig}$**	**BB$_{falsch}$**	**DV in %**	**DV in Pixel**
Augsburg	783	25	50,3	1219198
Nürnberg	1187	25	49,2	2023329
Bad Kissingen	1744	41	18,3	2131734
Windischeschenbach	1027	28	19,0	1293281

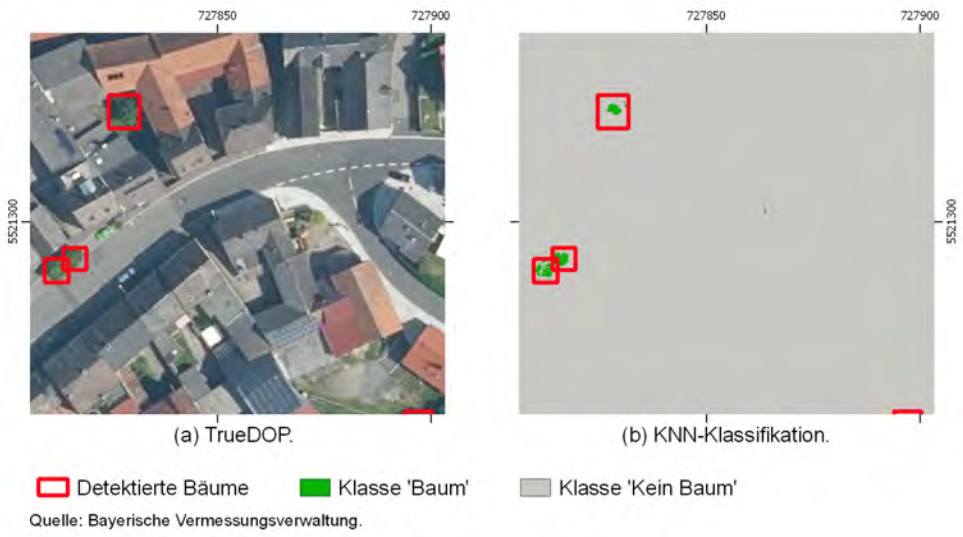


Abbildung 6.12: Das Detektionsergebnis des Modells yolov3$_{coco}$ nach der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Windischeschenbach. Der direkte Vergleich mit Abbildung 6.9 zeigt eine Verbesserung der Detektionsergebnisse.

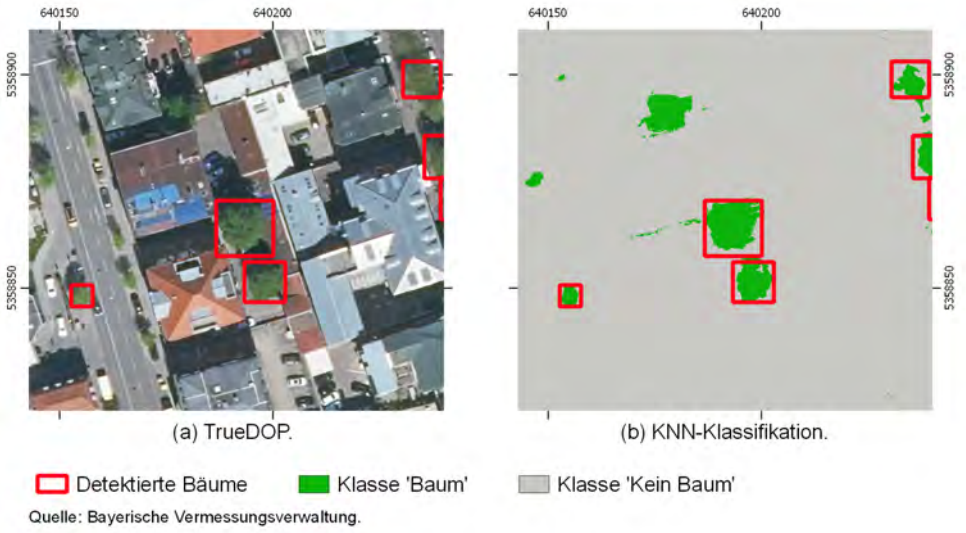


Abbildung 6.13: Das Detektionsergebnis des Modells yolov3$_{coco}$ nach der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Augsburg. Auch hier zeigt der direkte Vergleich mit Abbildung 6.10 zumindest eine Verbesserung bei der Fehldetektion.

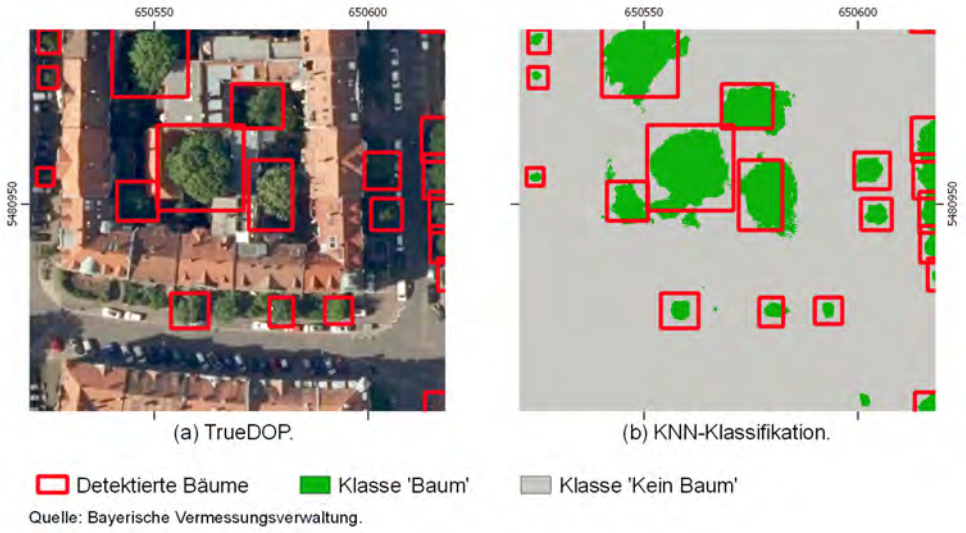


Abbildung 6.14: Das Detektionsergebnis des Modells yolov3$_{coco}$ nach der Domänenanpassung im Vergleich mit der KNN-Klassifikation sowie das TrueDOP des gleichen Bereichs im Untersuchungsgebiet Nürnberg. Die Verbesserung des Detektionsergebnisses nach der Domänenanpassung wird durch den Vergleich mit Abbildung 6.11 verdeutlicht.

7 Diskussion

In diesem Kapitel werden die Ergebnisse der durchgeführten Experimente diskutiert. Außerdem wird ein Ausblick gegeben, welche sinnvollen Verbesserungen vorgenommen werden können.

7.1 Einzelbaumdetektion

Das beste Ergebnis bei der Einzelbaumdetektion hat das Modell yolov3$_{coco}$ mit einem mAP Wert von 0,869 gezeigt (vgl. Tabelle 6.2).
Nach dem Vergleich der erreichten Präzisionen der Modelle mit zufälligen Gewichten und mit COCO-Gewichten lässt sich sagen, dass der verwendete Trainingsdatensatz allein nicht ausreichend war, um alle Gewichte der Netze optimal zu trainieren.
Allgemein deutet die hohe Trefferquote und die relativ kleine Präzision aller Modelle darauf hin, dass kein Modell eine eindeutige Definition der Klasse *Baum* erstellen konnte. Die Präzision des Modells yolov3$_{coco}$ von 0,725 bedeutet, dass über 20% aller Detektionen Fehldetektionen sind. Das kann auf das Rauschen im Trainingsdatensatz oder auf fehlende Baumbeispiele im Trainingsdatensatz zurückzuführen sein. Das Ergebnis könnte verbessert werden, wenn der Trainingsdatensatz um weitere Baumbeispiele erweitert werden würde.
Dieses Problem kann auch mit der Datenerweiterung (vgl. Abschnitt 2.1.2) angegangen werden. In diesem Projekt wurden Rotation, Skalierung und Spiegelung für eine Datenerweiterung verwendet. Diese Manipulationstechniken könnten durch eine Kontrastveränderung der Bilder ergänzt werden. Damit könnte das erstellte Modell robuster gegenüber kleinen Abweichungen in der Radiometrie der Luftbilder werden, die wegen unterschiedlichen Lichtverhältnissen bei der Aufnahme entstanden sein können.
Eine Verbesserung der Trefferquote bei dem Detektionsergebnis kann durch die Verwendung mehrerer Bilder mit unterschiedlicher Auflösung erzielt werden. Das Ergebnis eines Detektionsmodells kann abweichen, wenn es auf Bilder mit unterschiedlicher Auflösung angewendet wird (Ozge Unel u. a., 2019). Der Grund dafür ist der Unterschied in den Bildbereichen, die mit den Merkmalsfiltern gefaltet werden.

7.2 Domänenanpassung

Die Ergebnisse der Domänenanpassung mit dem Netz CycleGAN wurden sowohl durch die visuelle Betrachtung der übersetzten Bilder als auch anhand des Fréchet-Abstands ausgewertet. Bei der Auswertung der berechneten Fréchet-Abstände ist festzustellen, dass besonders bei der Anpassung des Nahinfrarotkanals gute Ergebnisse erreicht wurden. Gemäß der gewählten Metrik wurden die besten Ergebnisse bei der Domänenanpassung zwischen München und Nürnberg und Stockdorf und Windischeschenbach erzielt. Als Anhaltspunkt bei der Auswertung dient der Fréchet-Abstand zwischen München und Stockdorf, da diese Gebiete am selben Tag von der selben Befliegungsfirma aufgenommen wurden. Die vorhandenen Abweichungen zwischen diesen

Gebieten sind eventuell mit der unterschiedlichen Beschaffenheit zu erklären, da es sich einmal um ein städtisches und einmal um ein ländliches Gebiet handelt.
Bei der visuellen Betrachtung der Ergebnisse der Domänenanpassung kann die erfolgreiche Anpassung des Gebiets Nürnberg bestätigt werden (vgl. Abbildung 6.3). Im Vergleich dazu ist das Ergebnis der Domänenanpassung im Gebiet Bad Kissingen als mangelhaft zu bewerten. Das kann sowohl durch den berechneten Fréchet-Abstand nach der Domänenanpassung als auch durch die visuelle Auswertung der Ergebnisse bestätigt werden. Dieses Ergebnis kann mit einer höheren Komplexität der Szene in Bad Kissingen im Vergleich zu der Szene in Stockdorf begründet werden. Um das zu überprüfen, könnte die Domänenanpassung für dieses Gebiet mit einem erweiterten Datensatz, bei dem mehr Straßen und Infrastrukturobjekte enthalten sind, durchgeführt werden. Wie man in Abbildung 6.4 sehen kann, weist das Ergebnis der Übersetzung der Vegetation im Vergleich zu der Übersetzung der großen Straße kaum Fehler auf.
Der Fréchet-Abstand als Validierungsmetrik schafft eine Unterstützung bei der Bewertung der Ergebnisse der Domänenanpassung. Allgemein ist diese Metrik jedoch als kritisch zu betrachten, da sie nur einen Teil der Ergebnisse abbildet.

7.3 Validierung mit K-Nearest-Neighbors

In diesem Abschnitt werden die Ergebnisse der K-Nearest-Neighbors-Klassifikation sowie der Validierung der Detektionsergebnisse vor und nach der Domänenanpassung besprochen.

7.3.1 K-Nearest-Neighbors-Klassifikation

Für die K-Nearest-Neighbors-Klassifikation wurden große Trainingsdatensätze mit hochaufgelösten Fernerkundungsdaten verwendet, die viele Informationen über die spektralen und räumlichen Eigenschaften der Erdoberfläche enthalten. Sowohl die Gesamtgenauigkeit als auch der Kappa-Koeffizient der erstellten KNN-Modelle haben einen sehr hohen Wert von über 0,98 erreicht (vgl. Tabelle 6.5). Solche hohen Genauigkeiten können damit erklärt werden, dass alle KNN-Modelle jeweils nur für ein kleines Gebiet trainiert wurden. Bei der Klassifikation von kleinen Gebieten kann in der Regel eine bessere Genauigkeit als bei der Klassifikation von großen Gebieten erzielt werden (Hermosilla u. a., 2022).
Die Gesamtgenauigkeiten auf den Trainings- und den Testdatensätzen bei jedem Modell unterscheiden sich nur minimal. Das bedeutet, dass kein KNN-Modell an die Trainingsdaten überangepasst ist.
Die zusätzliche Korrektur der KNN-Klassifikation durch den Vergleich mit den nDOM Werten hat die Präzision des Klassifikationsergebnisses für die Klasse *Baum* verbessert. Das wird deutlich bei dem Vergleich der Konfusionsmatrizen vor und nach der Korrektur. Wie man in den Tabellen 6.7 bis 6.12 sieht, sinken fast in allen Fällen die FP Werte für die Klasse *Baum* nach der Korrektur. Diese positive Entwicklung des Klassifikationsergebnisses kann man auch in Abbildung 6.5 sehen. Das Ergebnis in (b) enthält viel mehr Rauschen in Form von falsch klassifizierten Einzelpixeln als das Ergebnis nach der Korrektur in (c).
Die steigenden FN Werte und dadurch auch die fallende Trefferquote nach der Korrektur sind damit zu erklären, dass die Bereiche für die Klasse *Baum* im Testdatensatz polygonweise ausgewählt wurden. Somit sind auch die Bereiche zwischen den Bäumen im Testdatensatz als Klasse *Baum* markiert. Nach der Korrektur werden diese Bereiche gegebenenfalls auf die Klasse *Kein*

Baum umgeschrieben, da sie nicht den Schwellenwert von 5 m erreichen. Damit das Korrekturergebnis durch diese Bereiche nicht verfälscht wird, könnten diese Bereiche vor der Berechnung der Korrektur aussortiert werden.
Wie man in Abbildung 6.6 sehen kann, gibt es auch Fehlklassifikationen, welche durch die durchgeführte Korrektur nicht verbessert werden konnten. Dabei handelt es sich vor allem um Fehlklassifikationen von begrünten Dachflächen. Diese Art der Fehlklassifikationen könnten durch einen zusätzlichen Abgleich mit dem Liegenschaftskataster korrigiert werden. Ähnlich wie bei der Korrektur des Klassifikationsergebnisses mit den nDOM Werten würden die Pixelwerte des Klassifikationsergebnisses der Klasse *Baum* auf die Klasse *Kein Baum* umgeschrieben werden, falls eine Überschneidung mit einem Gebäude laut Liegenschaftskataster vorliegt.

7.3.2 Validierung vor der Domänenanpassung

Das Validierungsergebnis zeigt, dass trotz der hohen Trefferquote des Detektionsmodells yolov3$_{coco}$ auf dem Testdatensatz in jedem Untersuchungsgebiet nur ein Teil der Vegetation mit einer Höhe von über 5 m detektiert werden konnte. Der Anteil der detektierten Vegetation an der Gesamtvegetation für jedes Untersuchungsgebiet ist in Tabelle 6.13 zu finden. Dass der Anteil der detektierten Vegetation nicht der Gesamtvegetation entspricht, wird nicht nur anhand der KNN-Klassifikation deutlich, sondern auch bei der visuellen Auswertung der Detektionsergebnisse. Ein Beispiel für fehlende Baumdetektionen ist in Abbildung 6.7 dargestellt. Bei den Untersuchungsgebieten, bei denen der Anteil der Gesamtvegetation größer ist, wurden tendenziell weniger Bäume detektiert. Das Problem der niedrigen Trefferquote kann mit den im Abschnitt 7.1 vorgeschlagenen Methoden angegangen werden.
Die Fehldetektionen des Detektionsmodells yolov3$_{coco}$ können generell in zwei Gruppen unterteilt werden.
Zu der ersten Gruppe gehören die Fehldetektionen der zu niedrigen Vegetation (vgl. Abbildung 6.8). Diese Art der Fehldetektionen wird als weniger kritisch bewertet und kann eventuell durch eine feinere Abstufung der nDOM Werte verbessert werden. So wird der Trainingsdatensatz mehr Höheninformation enthalten. Das könnte eine positive Auswirkung auf das Definieren der Klasse *Baum* durch das neuronale Netz haben.
Die zweite Fehlergruppe beinhaltet grobe Fehler bei den Detektionsergebnissen. Darunter sind Fehldetektionen von Nichtvegetationsobjekten wie z.B. Dachflächen und Solarzellen (vgl. Abbildungen 6.9 und 6.10). Dabei handelt es sich nicht um Fehldetektionen begrünter Dachflächen wie bei der KNN-Klassifikation. Der Grund dieser Fehler kann nicht auf eine fehlende Information für die Klasse *Hintergrund* zurückzuführen sein. Beispiele für diese Klasse sind im Trainingsdatensatz ausreichend vorhanden. Als mögliche Ursache kann die Abweichung in der Radiometrie dieser Untersuchungsgebiete von den Trainingsgebieten des Detektionsmodells in Betracht gezogen werden. Eine weitere Ursache für die Fehldetektion könnten fehlende Informationen aus dem Blaukanal der Spektralkanäle des TrueDOP sein. Dafür spricht die Tatsache, dass die gleichen Bereiche bei der KNN-Klassifikation richtig klassifiziert wurden. Für die KNN-Klassifikation wurde auch die Information aus dem Blaukanal der TrueDOP verwendet.
Eine Verbesserung kann eventuell durch die Verwendung des Verbesserten Vegetationsindex (Enhanced Vegetation Index, EVI) anstatt des normalisierten Vegetationsindexes (NDVI) erreicht werden. Im Vergleich zum NDVI berechnet sich dieser Index aus dem Verhältnis des Nahinfrarot-, Rot- und Blaukanals (EOS, 2022). Somit würde auch das Detektionsmodell mit der spektralen Information aus dem Blaukanal trainiert werden.

7.3.3 Validierung nach der Domänenanpassung

Trotz der erfolgreichen Domänenanpassung der Untersuchungsgebiete Augsburg und Windischeschenbach wurde keine Verbesserung bezüglich der Menge der detektierten Vegetation erreicht (vgl. Tabelle 6.14). Der Anteil der detektierten Vegetation an der Gesamtvegetation hat sich im Gebiet Windischeschenbach sogar verringert.
Im Gegensatz dazu hat sich der Anteil der detektierten Vegetation im Untersuchungsgebiet Nürnberg, welches am besten an die Quelldomäne angepasst wurde, fast um ein Drittel vergrößert. Beim Vergleich der Detektionsergebnisse vor der Domänenanpassung in Abbildung 6.11 und nach der Domänenanpassung in Abbildung 6.14 ist festzustellen, dass diese Verbesserung hauptsächlich durch eine präzisere Detektion der Grenzen einzelner Bäume erreicht wurde. Dafür spricht auch die fast gleich gebliebene Anzahl der Bounding Boxen bei beiden Ergebnissen.
Laut der berechneten Werte (BB_{falsch}) hat sich der Anteil der Fehldetektionen nur minimal in allen Gebieten verändert (vgl. Tabelle 6.14). Bei der visuellen Auswertung der Detektionsergebnisse lassen sich jedoch positive Veränderungen feststellen. Dies kann durch den Vergleich der Abbildungen 6.9 und 6.12, sowie 6.10 und 6.13 bestätigt werden.

8 Zusammenfassung

In dieser Arbeit wurde die Eignung von Deep-Learning-basierten Objektdetektionsmethoden, angewendet auf hochaufgelöste Fernerkundungsdaten, für die Erstellung eines Baumkatasters geprüft. Anhand der durchgeführten Experimente wurde untersucht, mit welchen Schwierigkeiten die Detektion einzelner Bäume verbunden sein kann.

Das Netz YOLOv3 hat bei der Detektion einzelner Bäume eine mittlere durchschnittliche Präzision von 0,869 auf dem Testdatensatz erreicht. Trotz der guten Ergebnisse auf dem Testdatensatz konnte jeweils nur ein Teil der Gesamtvegetation mit einer Höhe von über 5 m in den unabhängigen Validierungsgebieten detektiert werden. Der Anteil der detektierten Vegetation an der Gesamtvegetation liegt zwischen ca. 20% und 50%. Um dieses Verhalten zu analysieren, werden weitere Untersuchungen benötigt. Dabei kann z.B. untersucht werden, ob durch eine zusätzliche Datenerweiterung durch eine Kontrastveränderung und eine Verwendung der Bilder mit unterschiedlicher Auflösung bei der Detektion eine Verbesserung des Ergebnisses erzielt werden kann.

Die Ergebnisse der durchgeführten Domänenanpassung mit dem Netz CycleGAN sind allgemein als positiv zu bewerten. Die Fehler bei der Übersetzung einiger Gebiete sind damit zu erklären, dass die Ziel- und die Quelldomäne zu unterschiedlich in der Beschaffenheit der Gebiete sind. Das Ergebnis der Übersetzung kann verbessert werden, wenn die Übereinstimmung beider Domänen erhöht wird.
Die Domänenanpassung mit dem Netz CycleGAN ist eine gute Lösung für das Problem der Abweichungen in der Radiometrie der Luftbilder. Durch eine erfolgreiche Anpassung der Zieldomäne kann der Anteil der detektierten Vegetation an der Gesamtvegetation erhöht werden und Fehldetektionen verringert werden.

Die in diesem Projekt vorgeschlagene Methode für die Bewertung der Detektionsergebnisse durch den Vergleich mit der K-Nearest-Neighbors-Klassifikation ist als geeignet einzustufen. Trotz einiger Fehler bei der Klassifikation der Validierungsgebiete hat die KNN-Klassifikation ein gutes Ergebnis gezeigt. Dabei wurde bei der Klassifikation aller sechs Validierungsgebiete eine Gesamtgenauigkeit von über 0,98 erreicht. Die Qualität der Klassifikation kann weiter verbessert werden, indem das Klassifikationsergebnis zusätzlich mit den nDOM Werten und mit der Information aus dem Liegenschaftskataster abgeglichen wird. Allgemein kann mit dieser Bewertungsmethode das Problem fehlender Referenzen in Form von bekannten Baumstandorten angegangen werden.

Literaturverzeichnis

[Anaconda 2009] Anaconda: *Anaconda - By data scientists, for data scientists.* `https://www.anaconda.com/`. 2009. – Besucht am 2022-03-21

[Backhaus u. a. 2015] Backhaus, Klaus ; Erichson, Bernd ; Weiber, Rolf: *Multivariate Analysemethoden: eine anwendungsorientierte Einführung.* Springer-Verlag, 2015. – 461 S

[Benjdira u. a. 2019] Benjdira, Bilel ; Bazi, Yakoub ; Koubaa, Anis ; Ouni, Kais: Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images. In: *Remote Sensing* 11 (2019), Nr. 11, S. 1369

[Breiman 2001] Breiman, Leo: Random forests. In: *Machine learning* 45 (2001), Nr. 1, S. 5–32

[Carreiras u. a. 2017] Carreiras, Joã. ; Jones, Joshua ; Lucas, Richard M. ; Shimabukuro, Yosio E.: Mapping major land cover types and retrieving the age of secondary forests in the Brazilian Amazon by combining single-date optical and radar remote sensing data. In: *Remote Sensing of Environment* 194 (2017), S. 16–32

[COCO 2019] COCO: *COCO 2019 Object Detection Task.* `https://cocodataset.org/#detection-2019`. 2019. – Besucht am 2022-03-25

[Cohen 1960] Cohen, Jacob: A coefficient of agreement for nominal scales. In: *Educational and psychological measurement* 20 (1960), Nr. 1, S. 37–46

[Congalton 2001] Congalton, Russell G.: Accuracy assessment and validation of remotely sensed and other spatial information. In: *International Journal of Wildland Fire* 10 (2001), Nr. 4, S. 321–328

[Culman u. a. 2020] Culman, María ; Delalieux, Stephanie ; Tricht, Kristof V.: Individual Palm Tree Detection Using Deep Learning on RGB Imagery to Support Tree Inventory. In: *Remote Sensing* 12 (2020), Nr. 21, S. 3476

[De Lange 2013] De Lange, Norbert: *Geoinformatik: in Theorie und Praxis.* Springer-Verlag, 2013. – 460 S

[Diawara 2019] Diawara, Norou: *Modern Statistical Methods for Spatial and Multivariate Data.* Springer, 2019. – 115 S

[Dowson und Landau 1982] Dowson, DC ; Landau, BV666017: The Fréchet distance between multivariate normal distributions. In: *Journal of multivariate analysis* 12 (1982), Nr. 3, S. 450–455

[EOS 2022] EOS: *EOS Earth Observing System - Vegetation Index For Agriculture In Digital Solutions.* `https://eos.com/blog/vegetation-indices/`. 2022. – Besucht am 2022-04-21

[EOS Data Analytics 2022] EOS Data Analytics: *Color Infrared (Vegetation).* `https://eos.com/color-infrared/`. 2022. – Besucht am 2022-20-03

[ESA 2022] ESA: *European Space Agency - Sentinel-2 Mission*. https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/sentinel-2. 2022. – Besucht am 2022-03-19

[Everingham u. a. 2015] Everingham, Mark ; Eslami, SM ; Van Gool, Luc ; Williams, Christopher K. ; Winn, John ; Zisserman, Andrew: The pascal visual object classes challenge: A retrospective. In: *International journal of computer vision* 111 (2015), Nr. 1, S. 98–136

[Fehrmann und Kleinn 2007] Fehrmann, Flutz ; Kleinn, Christoph: A k-nearest neighbor approach for estimation of single-tree biomass. In: *In: McRoberts, Ronald E.; Reams, Gregory A.; Van Deusen, Paul C.; McWilliams, William H., eds. Proceedings of the seventh annual forest inventory and analysis symposium; October 3-6, 2005; Portland, ME. Gen. Tech. Rep. WO-77. Washington, DC: US Department of Agriculture, Forest Service: 251-259.* Bd. 77, 2007

[Foster 2019] Foster, David: *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media, 2019

[Gareth u. a. 2013] Gareth, James ; Daniela, Witten ; Trevor, Hastie ; Robert, Tibshirani: *An introduction to statistical learning: with applications in R*. Spinger, 2013

[GDAL 1998] GDAL: *GDAL - Geospatial Data Abstraction Library*. https://gdal.org/. 1998. – Besucht am 2022-03-21

[Girshick u. a. 2014] Girshick, Ross ; Donahue, Jeff ; Darrell, Trevor ; Malik, Jitendra: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, S. 580–587

[Glorot und Bengio 2010] Glorot, Xavier ; Bengio, Yoshua: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics* JMLR Workshop and Conference Proceedings (Veranst.), 2010, S. 249–256

[Goodfellow u. a. 2016] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep learning*. MIT press, 2016

[Goodfellow u. a. 2014] Goodfellow, Ian ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: Generative adversarial nets. In: *Advances in neural information processing systems* 27 (2014)

[Guo u. a. 2018] Guo, Yanhui ; Han, Siming ; Li, Ying ; Zhang, Cuifen ; Bai, Yu: K-Nearest Neighbor combined with guided filter for hyperspectral image classification. In: *Procedia Computer Science* 129 (2018), S. 159–165

[Haas 2019] Haas, Amelie: Nutzung künstlicher Neuronaler Netze zur Detektion von Stadtgrün. (2019)

[He u. a. 2013] He, K. ; Sun, J. ; Tang, X.: Guided Image Filtering. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), Nr. 6, S. 1397–1409

[Henon 2018] Henon, Yann: *Pytorch implementation of RetinaNet*. https://github.com/yhenon/pytorch-retinanet. 2018

[Hermosilla u. a. 2022] Hermosilla, Txomin ; Wulder, Michael A. ; White, Joanne C. ; Coops, Nicholas C.: Land cover classification in an era of big and open data: Optimizing localized implementation and training data selection to improve mapping outcomes. In: *Remote Sensing of Environment* 268 (2022), S. 112780

[Heusel u. a. 2017] Heusel, Martin ; Ramsauer, Hubert ; Unterthiner, Thomas ; Nessler, Bernhard ; Hochreiter, Sepp: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: *Advances in neural information processing systems* 30 (2017)

[Huang u. a. 2019] Huang, Zhongling ; Dumitru, Corneliu O. ; Pan, Zongxu ; Lei, Bin ; Datcu, Mihai: Can a Deep Network Understand the Land Cover Across Sensors? In: *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium* IEEE (Veranst.), 2019, S. 9847–9850

[Isola u. a. 2017] Isola, Phillip ; Zhu, Jun-Yan ; Zhou, Tinghui ; Efros, Alexei A.: Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 1125–1134

[JAXA 2022] JAXA: *Japan Aerospace Exploration Agency. Earth Observation Research Center: ALOS - PALSAR*. `https://www.eorc.jaxa.jp/ALOS/en/alos/sensor/palsar_e.htm`. 2022. – Besucht am 2022-03-19

[Jiang u. a. 2019] Jiang, Xiaoyue ; Hadid, Abdenour ; Pang, Yanwei ; Granger, Eric ; Feng, Xiaoyi: *Deep Learning in object detection and recognition*. Springer, 2019

[Jocher 2020] Jocher, Glenn: *ultralytics/yolov3: v9.5.0 - YOLOv5 v5.0 release compatibility update for YOLOv3*. `https://github.com/ultralytics/yolov3`. 2020

[Ketkar und Moolayil 2021] Ketkar, Nikhil ; Moolayil, Jojo: *Deep learning with Python: learn best practices of deep learning models with PyTorch*. Springer, 2021

[Krizhevsky u. a. 2009] Krizhevsky, Alex ; Hinton, Geoffrey u. a.: Learning multiple layers of features from tiny images. (2009)

[LabelImg 2015] LabelImg: *LabelImg - a graphical image annotation tool*. `https://github.com/tzutalin/labelImg`. 2015. – Besucht am 2022-03-21

[LDBV 2022] LDBV: *ATKIS - Amtliches Digitales Basis-Landschaftsmodell*. `https://www.ldbv.bayern.de/produkte/atkis-basis-dlm.html`. 2022. – Besucht am 2022-03-19

[LeakyReLU 2022] LeakyReLU: *LeakyRELU Activation Function*. `https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html`. 2022. – Besucht am 2022-03-28

[LeCun u. a. 1989] LeCun, Yann ; Boser, Bernhard ; Denker, John ; Henderson, Donnie ; Howard, Richard ; Hubbard, Wayne ; Jackel, Lawrence: Handwritten digit recognition with a back-propagation network. In: *Advances in neural information processing systems* 2 (1989)

[LeCun u. a. 1998] LeCun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[Lin u. a. 2017a] Lin, Tsung-Yi ; Dollár, Piotr ; Girshick, Ross ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge: Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 2117–2125

[Lin u. a. 2017b] Lin, Tsung-Yi ; Goyal, Priya ; Girshick, Ross ; He, Kaiming ; Dollár, Piotr: Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*, 2017, S. 2980–2988

[Lin u. a. 2014] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Dollár, Piotr ; Zitnick, C L.: Microsoft coco: Common objects in context. In: *European conference on computer vision* Springer (Veranst.), 2014, S. 740–755

[Liu u. a. 2016] Liu, Wei ; Anguelov, Dragomir ; Erhan, Dumitru ; Szegedy, Christian ; Reed, Scott ; Fu, Cheng-Yang ; Berg, Alexander C.: Ssd: Single shot multibox detector. In: *European conference on computer vision* Springer (Veranst.), 2016, S. 21–37

[LWF 2012] LWF: *Bayerisches Staatsministerium für Ernährung, Landwirtschaft und Forsten: Waldfläche*. `https://www.lwf.bayern.de/bwi/080772/index.php?layer=rss`. 2012. – Besucht am 2022-03-19

[Michelucci 2019] Michelucci, Umberto: *Advanced applied deep learning: convolutional neural networks and object detection*. Springer, 2019

[Mirza und Osindero 2014] Mirza, Mehdi ; Osindero, Simon: Conditional generative adversarial nets. In: *arXiv preprint arXiv:1411.1784* (2014)

[NASA 2022] NASA: *Jet Propulsion Laboratory. AVIRIS - Airborne Visible/InfraRed Imaging Spectrometer*. `https://aviris.jpl.nasa.gov`. 2022. – Besucht am 2022-03-19

[Nvidia Developer 2022] Nvidia Developer: *Deep Learning*. `https://developer.nvidia.com/deep-learning`. 2022. – Besucht am 2022-03-16

[Ozge Unel u. a. 2019] Ozge Unel, F ; Ozkalayci, Burak O. ; Cigla, Cevahir: The power of tiling for small object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, S. 0–0

[Padilla u. a. 2020] Padilla, Rafael ; Netto, Sergio L. ; Da Silva, Eduardo A.: A survey on performance metrics for object-detection algorithms. In: *2020 international conference on systems, signals and image processing (IWSSIP)* IEEE (Veranst.), 2020, S. 237–242

[Padilla u. a. 2021] Padilla, Rafael ; Passos, Wesley L. ; Dias, Thadeu L. ; Netto, Sergio L. ; Silva, Eduardo A. da: A comparative analysis of object detection metrics with a companion open-source toolkit. In: *Electronics* 10 (2021), Nr. 3, S. 279

[Petersen u. a. 2008] Petersen, Jörg ; Dassau, Otto ; Dauck, Hans-Peter ; Janinhoff, Nicole: Angewandte Vegetationskartierung großräumiger Projektgebiete auf Basis digitaler Luftbilddaten – eine kombinierte Methode aus Fernerkundung, GIS und nahezu flächendeckender Verifizierung im Gelände. In: *AGIT Symposium für Angewandte Geoinformatik* (2008)

[Python 1991] Python: *Python is a programming language that lets you work quickly and integrate systems more effectively*. `https://www.python.org/`. 1991. – Besucht am 2022-03-21

[PyTorch 2016] PyTorch: *PyTorch - End-to-End Machine Learning Framework*. `https://pytorch.org/`. 2016. – Besucht am 2022-03-21

[QGIS 2002] QGIS: *QGIS - Ein freies Open-Source-Geographisches-Informationssystem.* `https://www.qgis.org/de/site/index.html`. 2002. – Besucht am 2022-03-21

[Radford u. a. 2015] Radford, Alec ; Metz, Luke ; Chintala, Soumith: Unsupervised representation learning with deep convolutional generative adversarial networks. In: *arXiv preprint arXiv:1511.06434* (2015)

[Redmon u. a. 2016] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 779–788

[Redmon und Farhadi 2018] Redmon, Joseph ; Farhadi, Ali: Yolov3: An incremental improvement. In: *arXiv preprint arXiv:1804.02767* (2018)

[Ren u. a. 2015] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster r-cnn: Towards real-time object detection with region proposal networks. In: *arXiv preprint arXiv:1506.01497* (2015)

[Roettger 2007] Roettger, Stefan: NDVI-based vegetation rendering. In: *Computer Graphics and Imaging CGIM* 7 (2007)

[Russakovsky u. a. 2015] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael u. a.: Imagenet large scale visual recognition challenge. In: *International journal of computer vision* 115 (2015), Nr. 3, S. 211–252

[Salimans u. a. 2016] Salimans, Tim ; Goodfellow, Ian ; Zaremba, Wojciech ; Cheung, Vicki ; Radford, Alec ; Chen, Xi: Improved techniques for training gans. In: *Advances in neural information processing systems* 29 (2016)

[Santos u. a. 2019] Santos, Anderson Aparecido d. ; Marcato Junior, Jose ; Araújo, Márcio S. ; Di Martini, David R. ; Tetila, Everton C. ; Siqueira, Henrique L. ; Aoki, Camila ; Eltner, Anette ; Matsubara, Edson T. ; Pistori, Hemerson u. a.: Assessment of CNN-based methods for individual tree detection on images captured by RGB cameras attached to UAVs. In: *Sensors* 19 (2019), Nr. 16, S. 3595

[Scikit-learn 2010] Scikit-learn: *Scikit-learn: Machine Learning in Python.* `https://scikit-learn.org/stable/`. 2010. – Besucht am 2022-03-21

[SEOS 2012] SEOS: *Einführung in die Kategorisierung von Objekten anhand ihrer Daten.* `https://seos-project.eu/classification/classification-c01-p05.de.html`. 2012. – Besucht am 2022-03-22

[Süße und Rodner 2014] Süsse, Herbert ; Rodner, Erik: *Bildverarbeitung und Objekterkennung.* Springer, 2014

[SWR 2022] SWR: *Schutzgemeinschaft Deutscher Wald, Bundesverband e.V.: Bäume in der Stadt.* `https://www.sdw.de/ueber-den-wald/waldwissen/baeume-in-der-stadt/`. 2022. – Besucht am 2022-03-05

[Taha und Hanbury 2015] Taha, Abdel A. ; Hanbury, Allan: Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. In: *BMC medical imaging* 15 (2015), Nr. 1, S. 1–28

[Tenikl u. a. 2019] Tenikl, Julia ; Wurm, Michael ; Weigand, Matthias ; Staab, Jeroen ; Müller, Inken ; Taubenböck, Hannes: Satellitengestützte Vermessung von städtischem Grün in deutschen Städten. In: *11. Dresdner Flächennutzungssymposium* (2019)

[Weidman 2019] Weidman, Seth: *Deep Learning from Scratch: Building with Python from First Principles*. O'Reilly Media, 2019

[Weier und Herring 2000] Weier, J ; Herring, D: Measuring vegetation (NDVI & EVI). Earth observatory. In: *National Aeronautics and Space Administration* (2000)

[Zhang u. a. 2020] Zhang, Guangyuan ; Rui, Xiaoping ; Poslad, Stefan ; Song, Xianfeng ; Fan, Yonglei ; Wu, Bang: A method for the estimation of finely-grained temporal spatial human population density distributions based on cell phone call detail records. In: *Remote Sensing* 12 (2020), Nr. 16, S. 2572

[Zhu u. a. 2017] Zhu, Jun-Yan ; Park, Taesung ; Isola, Phillip ; Efros, Alexei A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*, 2017, S. 2223–2232

[Zhu u. a. 2018] Zhu, Jun-Yan ; Park, Taesung ; Wang, Tongzhou: *CycleGAN and pix2pix in PyTorch*. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`. 2018